NAME

intro – file formats used or read by various programs

DESCRIPTION

This section describes formats of files used by various programs.

A 5V section number means one or more of the following:

- The man page documents System V formats only.
- The man page documents default SunOS formats, and System V formats as they differ from the default formats. These System V differences are presented under **SYSTEM V** section headers.
- The man page documents formats compliant with *IEEE Std 1003.1-1988* (POSIX.1).

LIST OF FILE FORMATS

Name	Appears on Page	Description
acct	**acct**(5)	execution accounting file
addresses	**aliases**(5)	addresses and aliases for sendmail
aliases	**aliases**(5)	addresses and aliases for sendmail
a.out	**a.out**(5)	assembler and link editor output format
ar	**ar**(5)	archive (library) file format
audit_control	**audit_control**(5)	control information for system audit daemon
audit_data	**audit_data**(5)	current information on audit daemon
audit.log	**audit.log**(5)	the security audit trail file
auto.home	**auto.home**(5)	automount map for home directories
auto.vol	**auto.vol**(5)	automount map for volumes
bar	**bar**(5)	tape archive file format
boards.pc	**boards.pc**(5)	AT- and XT-compatible boards for DOS windows
bootparams	**bootparams**(5)	boot parameter data base
bootservers	**bootservers**(5)	NIS bootservers file
coff	**coff**(5)	common assembler and link editor output
core	**core**(5)	format of memory image file
cpio	**cpio**(5)	format of cpio archive
crontab	**crontab**(5)	table of times to run periodic jobs
dir	**dir**(5)	format of directories
dump	**dump**(5)	incremental dump format
dumpdates	**dump**(5)	incremental dump format
environ	**environ**(5V)	user environment
ethers	**ethers**(5)	Ethernet address to hostname database or NIS domain
exports	**exports**(5)	directories to export to NFS clients
ext_ports	**ext_ports**(5)	external ports file for network printers, terminals, and modems
fbtab	**fbtab**(5)	framebuffer table
fcntl	**fcntl**(5)	file control options
forward	**aliases**(5)	addresses and aliases for sendmail
fs	**fs**(5)	format of a 4.2 (ufs) file system volume
fspec	**fspec**(5)	format specification in text files
fstab	**fstab**(5)	static filesystem mounting table, mounted filesystems table
ftpusers	**ftpusers**(5)	list of users prohibited by FTP
gettytab	**gettytab**(5)	terminal configuration data base
group	**group**(5)	group file
group.adjunct	**group.adjunct**(5)	group security data file
help	**help**(5)	help file format
help_viewer	**help_viewer**(5)	help viewer file format
hosts	**hosts**(5)	host name data base
hosts.equiv	**hosts.equiv**(5)	trusted hosts by system and by user

indent.pro	**indent.pro**(5)	default options for indent
inetd.conf	**inetd.conf**(5)	Internet servers database
inode	**fs**(5)	format of a 4.2 (ufs) file system volume
internat	**internat**(5)	key mapping table for internationalization
ipalloc.netrange	**ipalloc.netrange**(5)	range of addresses to allocate
keytables	**keytables**(5)	keyboard table descriptions for loadkeys and dumpkeys
lastlog	**utmp**(5V)	login records
link	**link**(5)	link editor interfaces
locale	**locale**(5)	locale database
magic	**magic**(5)	file command's magic number file
mtab	**fstab**(5)	static filesystem mounting table, mounted filesystems table
mtab	**mtab**(5)	mounted file system table
netgroup	**netgroup**(5)	list of network groups
netmasks	**netmasks**(5)	network mask data base
netrc	**netrc**(5)	file for ftp remote login data
networks	**networks**(5)	network name data base
orgrc	**orgrc**(5)	organizer configuration and initialization file
passwd	**passwd**(5)	password file
passwd.adjunct	**passwd.adjunct**(5)	user security data file
phones	**phones**(5)	remote host phone number data base
plot	**plot**(5)	graphics interface
pnp.sysnames	**pnp.sysnames**(5)	file used to allocate system names
policies	**policies**(5)	network administration policies
printcap	**printcap**(5)	printer capability data base
proto	**proto**(5)	prototype job file for at
protocols	**protocols**(5)	protocol name data base
publickey	**publickey**(5)	public key database
queuedefs	**queuedefs**(5)	queue description file for at, batch, and cron
rasterfile	**rasterfile**(5)	Sun's file format for raster images
remote	**remote**(5)	remote host description file
resolv.conf	**resolv.conf**(5)	configuration file for domain name system resolver
rfmaster	**rfmaster**(5)	Remote File Sharing name server master file
rgb	**rgb**(5)	available colors (by name) for coloredit
rhosts	**hosts.equiv**(5)	trusted hosts by system and by user
rootmenu	**rootmenu**(5)	root menu specification for SunView
rpc	**rpc**(5)	rpc program number data base
sccsfile	**sccsfile**(5)	format of an SCCS history file
services	**services**(5)	Internet services and aliases
setup.pc	**setup.pc**(5)	master configuration file for DOS
sm	**sm**(5)	in.statd directory and file structures
sm	**statmon**(5)	statd directories and file structures
sm.bak	**sm**(5)	in.statd directory and file structures
sm.bak	**statmon**(5)	statd directories and file structures
sm.state	**sm**(5)	in.statd directory and file structures
state	**statmon**(5)	statd directories and file structures
sunview	**sunview**(5)	initialization file for SunView
svdtab	**svdtab**(5)	SunView device table
syslog.conf	**syslog.conf**(5)	configuration file for syslogd system log daemon
systems	**systems**(5)	NIS systems file
tar	**tar**(5)	tape archive file format
termcap	**termcap**(5)	terminal capability data base
term	**term**(5)	terminal driving tables for nroff
term	**term**(5V)	format of compiled term file

terminfo	**terminfo**(5V)	terminal capability data base
toc	**toc**(5)	table of contents of optional clusters
translate	**translate**(5)	input and output files for system message translation
ttys	**ttytab**(5)	terminal initialization data
ttytab	**ttytab**(5)	terminal initialization data
types	**types**(5)	primitive system data types
tzfile	**tzfile**(5)	time zone information
ugid_alloc.range	**ugid_alloc.range**(5)	range of user IDs and group IDs to allocate
updaters	**updaters**(5)	configuration file for NIS updating
utmp	**utmp**(5V)	login records
uuencode	**uuencode**(5)	format of an encoded uuencode file
vfont	**vfont**(5)	font formats
vgrindefs	**vgrindefs**(5)	vgrind's language definition data base
wtmp	**utmp**(5V)	login records
xtab	**exports**(5)	directories to export to NFS clients
ypaliases	**ypaliases**(5)	NIS aliases for sendmail
ypfiles	**ypfiles**(5)	NIS database and directory structure
ypgroup	**ypgroup**(5)	NIS group file
yppasswd	**yppasswd**(5)	NIS password file
ypprintcap	**ypprintcap**(5)	NIS printer capability database

NAME

a.out – assembler and link editor output format

SYNOPSIS

```
#include <a.out.h>
#include <stab.h>
#include <nlist.h>
```

AVAILABILITY

Sun-2, Sun-3, and Sun-4 systems only. For Sun386i systems refer to **coff**(5).

DESCRIPTION

a.out is the output format of the assembler **as**(1) and the link editor **ld**(1). The link editor makes **a.out** executable files.

A file in **a.out** format consists of: a header, the program text, program data, text and data relocation information, a symbol table, and a string table (in that order). In the header, the sizes of each section are given in bytes. The last three sections may be absent if the program was loaded with the –s option of **ld** or if the symbols and relocation have been removed by **strip**(1).

The machine type in the header indicates the type of hardware on which the object code can be executed. Sun-2 code runs on Sun-3 systems, but not vice versa. Program files predating release 3.0 are recognized by a machine type of '**0**'. Sun-4 code may not be run on Sun-2 or Sun-3, nor vice versa.

Header

The header consists of a **exec** structure. The **exec** structure has the form:

```
struct exec {
        unsigned char   a_dynamic:1;    /* has a __DYNAMIC */
        unsigned char   a_toolversion:7; /* version of toolset used to create this file */
        unsigned char   a_machtype;     /* machine type */
        unsigned short  a_magic;        /* magic number */
        unsigned long   a_text;         /* size of text segment */
        unsigned long   a_data;         /* size of initialized data */
        unsigned long   a_bss;          /* size of uninitialized data */
        unsigned long   a_syms;         /* size of symbol table */
        unsigned long   a_entry;        /* entry point */
        unsigned long   a_trsize;       /* size of text relocation */
        unsigned long   a_drsize;       /* size of data relocation */
};
```

The members of the structure are:

a_dynamic **1** if the **a.out** file is dynamically linked or is a shared object, **0** otherwise.

a_toolversion The version number of the toolset (**as**, **ld**, etc.) used to create the file.

a_machtype One of the following:

0	pre-3.0 executable image
M_68010	executable image using only MC68010 instructions that can run on Sun-2 or Sun-3 systems.
M_68020	executable image using MC68020 instructions that can run only on Sun-3 systems.
M_SPARC	executable image using SPARC instructions that can run only on Sun-4 systems.

a_magic One of the following:

OMAGIC	An text executable image which is not to be write-protected, so the data segment is immediately contiguous with the text segment.

NMAGIC A write-protected text executable image. The data segment begins at the first segment boundary following the text segment, and the text segment is not writable by the program. When the image is started with **execve**(2V), the entire text and data segments will be read into memory.

ZMAGIC A page-aligned text executable image. the data segment begins at the first segment boundary following the text segment, and the text segment is not writable by the program. The text and data sizes are both multiples of the page size, and the pages of the file will be brought into the running image as needed, and not pre-loaded as with the other formats. This is the default format produced by **ld**(1).

The macro **N_BADMAG** takes an **exec** structure as an argument; it evaluates to 1 if the **a_magic** field of that structure is invalid, and evaluates to 0 if it is valid.

a_text The size of the text segment, in bytes.

a_data The size of the initialized portion of the data segment, in bytes.

a_bss The size of the "uninitialized" portion of the data segment, in bytes. This portion is actually initialized to zero. The zeroes are not stored in the **a.out** file; the data in this portion of the data segment is zeroed out when it is loaded.

a_syms The size of the symbol table, in bytes.

a_entry The virtual address of the entry point of the program; when the image is started with **execve**, the first instruction executed in the image is at this address.

a_trsize The size of the relocation information for the text segment.

a_drsize The size of the relocation information for the data segment.

The macros **N_TXTADDR**, **N_DATADDR**, and **N_BSSADDR** give the memory addresses at which the text, data, and bss segments, respectively, will be loaded.

In the **ZMAGIC** format, the size of the header is included in the size of the text section; in other formats, it is not.

When an **a.out** file is executed, three logical segments are set up: the text segment, the data segment (with uninitialized data, which starts off as all 0, following initialized data), and a stack. For the **ZMAGIC** format, the header is loaded with the text segment; for other formats it is not.

Program execution begins at the address given by the value of the **a_entry** field.

The stack starts at the highest possible location in the memory image, and grows downwards. The stack is automatically extended as required. The data segment is extended as requested by **brk**(2) or **sbrk**.

Text and Data Segments

The text segment begins at the start of the file for **ZMAGIC** format, or just after the header for the other formats. The **N_TXTOFF** macro returns this absolute file position when given an **exec** structure as argument. The data segment is contiguous with the text and immediately followed by the text relocation and then the data relocation information. The **N_DATOFF** macro returns the absolute file position of the beginning of the data segment when given an **exec** structure as argument.

Relocation

The relocation information appears after the text and data segments. The **N_TRELOFF** macro returns the absolute file position of the relocation information for the text segment, when given an **exec** structure as argument. The **N_DRELOFF** macro returns the absolute file position of the relocation information for the data segment, when given an **exec** structure as argument. There is no relocation information if **a_trsize+a_drsize==0**.

Relocation (Sun-2 and Sun-3 Systems)

If a byte in the text or data involves a reference to an undefined external symbol, as indicated by the relocation information, then the value stored in the file is an offset from the associated external symbol. When

the file is processed by the link editor and the external symbol becomes defined, the value of the symbol is added to the bytes in the file. If a byte involves a reference to a relative location, or relocatable segment, then the value stored in the file is an offset from the associated segment.

If relocation information is present, it amounts to eight bytes per relocatable datum as in the following structure:

```
struct reloc_info_68k {
    long        r_address;          /* address which is relocated */
unsigned int    r_symbolnum:24,     /* local symbol ordinal */
    r_pcrel:1,          /* was relocated pc relative already */
    r_length:2,         /* 0=byte, 1=word, 2=long */
    r_extern:1,         /* does not include value of sym referenced */
    r_baserel:1,        /* linkage table relative */
    r_jmptable:1,       /* pc-relative to jump table */
    r_relative:1,       /* relative relocation */
    :1;
};
```

If **r_extern** is 0, then **r_symbolnum** is actually an **n_type** for the relocation (for instance, **N_TEXT** meaning relative to segment text origin.)

Relocation (Sun-4 System)

If a byte in the text or data involves a reference to an undefined external symbol, as indicated by the relocation information, then the value stored in the file is ignored. Unlike the Sun-2 and Sun-3 system, the offset from the associated symbol is kept with the relocation record. When the file is processed by the link editor and the external symbol becomes defined, the value of the symbol is added to this offset, and the sum is inserted into the bytes in the text or data segment.

If relocation information is present, it amounts to twelve bytes per relocatable datum as in the following structure:

```
enum reloc_type
{
    RELOC_8,            RELOC_16,           RELOC_32,           /* simplest relocs */
    RELOC_DISP8,        RELOC_DISP16,       RELOC_DISP32,       /* Disp's (pc-rel) */
    RELOC_WDISP30,      RELOC_WDISP22,      /* SR word disp's */
    RELOC_HI22,         RELOC_22,           /* SR 22-bit relocs */
    RELOC_13,           RELOC_LO10,         /* SR 13&10-bit relocs */
    RELOC_SFA_BASE,     RELOC_SFA_OFF13,    /* SR S.F.A. relocs */
    RELOC_BASE10,       RELOC_BASE13,       RELOC_BASE22,       /* base_relative pic */
    RELOC_PC10,         RELOC_PC22,         /* special pc-rel pic*/
    RELOC_JMP_TBL,      /* jmp_tbl_rel in pic */
    RELOC_SEGOFF16,     /* ShLib offset-in-seg */
    RELOC_GLOB_DAT,     RELOC_JMP_SLOT,     RELOC_RELATIVE,     /* rtld relocs */
};

struct reloc_info_sparc   /* used when header.a_machtype == M_SPARC */
{
    unsigned long int   r_address;          /* relocation addr (offset in segment) */
    unsigned int        r_index     :24;    /* segment index or symbol index */
    unsigned int        r_extern    : 1;    /* if F, r_index==SEG#; if T, SYM idx */
    int                 : 2;                /* <unused> */
    enum reloc_type     r_type      : 5;    /* type of relocation to perform */
    long int            r_addend;           /* addend for relocation value */
};
```

If **r_extern** is 0, then **r_index** is actually a **n_type** for the relocation (for instance, **N_TEXT** meaning relative to segment text origin.)

Symbol Table

The **N_SYMOFF** macro returns the absolute file position of the symbol table when given an **exec** structure as argument. Within this symbol table, distinct symbols point to disjoint areas in the string table (even when two symbols have the same name). The string table immediately follows the symbol table; the **N_STROFF** macro returns the absolute file position of the string table when given an **exec** structure as argument. The first 4 bytes of the string table are not used for string storage, but rather contain the size of the string table. This size *includes* the 4 bytes; thus, the minimum string table size is 4. Layout information as given in the include file for the Sun system is shown below.

The layout of a symbol table entry and the principal flag values that distinguish symbol types are given in the include file as follows:

```
struct nlist {
        union {
                char    *n_name;        /* for use when in-memory */
                long    n_strx;         /* index into file string table */
        } n_un;
        unsigned char   n_type;         /* type flag, that is, N_TEXT etc; see below */
        char    n_other;
        short   n_desc;                 /* see <stab.h> */
        unsigned        n_value;        /* value of this symbol (or adb offset) */
};
#define n_hash  n_desc                  /* used internally by ld */
/*
* Simple values for n_type.
*/
#define N_UNDF  0x0                     /* undefined */
#define N_ABS   0x2                     /* absolute */
#define N_TEXT  0x4                     /* text */
#define N_DATA  0x6                     /* data */
#define N_BSS   0x8                     /* bss */
#define N_COMM  0x12                    /* common (internal to ld) */
#define N_FN    0x1f                    /* file name symbol */
#define N_EXT   01                      /* external bit, or'ed in */
#define N_TYPE  0x1e                    /* mask for all the type bits */
/*
* Other permanent symbol table entries have some of the N_STAB bits set.
* These are given in <stab.h>
*/
#define N_STAB  0xe0                    /* if any of these bits set, don't discard */
```

In the **a.out** file a symbol's **n_un.n_strx** field gives an index into the string table. A **n_strx** value of 0 indicates that no name is associated with a particular symbol table entry. The field **n_un.n_name** can be used to refer to the symbol name only if the program sets this up using **n_strx** and appropriate data from the string table. Because of the union in the **nlist** declaration, it is impossible in C to statically initialize such a structure. If this must be done (as when using **nlist**(3V)) include the file **<nlist.h>**, rather than **<a.out.h>**. This contains the declaration without the union.

If a symbol's type is undefined external, and the value field is non-zero, the symbol is interpreted by the loader **ld** as the name of a common region whose size is indicated by the value of the symbol.

SEE ALSO

adb(1), **as**(1), **cc**(1V), **dbx**(1), **ld**(1), **nm**(1), **strip**(1), **brk**(2), **nlist**(3V), **coff**(5)

NAME

acct – execution accounting file

SYNOPSIS

#include <sys/acct.h>

DESCRIPTION

The **acct**(2V) system call makes entries in an accounting file for each process that terminates. The accounting file is a sequence of entries whose layout, as defined by the include file is:

```
typedef u_short comp_t;

struct acct
{
        char    ac_flag;        /* Accounting flag */
        char    ac_stat;        /* Exit status */
        uid_t   ac_uid;                 /* Accounting user ID */
        gid_t   ac_gid;                 /* Accounting group ID */
        dev_t   ac_tty;                 /* control typewriter */
        time_t  ac_btime;               /* Beginning time */
        comp_t  ac_utime;               /* Accounting user time */
        comp_t  ac_stime;               /* Accounting system time */
        comp_t  ac_etime;               /* Accounting elapsed time */
        comp_t  ac_mem;                         /* average memory usage */
        comp_t  ac_io;                  /* chars transferred */
        comp_t  ac_rw;                  /* blocks read or written */
        char    ac_comm[8];             /* Accounting command name */
};
```

The type **comp_t** is a 3 bits base 8 exponent, 13 bit fraction "floating point" number. If the process does an **execve**(2V), the first 8 characters of the filename appear in **ac_comm**. **ac_flag** contains bits indicating whether **execve**(2V) was ever accomplished, and whether the process ever had super-user privileges.

SEE ALSO

acct(2V), **execve**(2V), sa(8)

NAME

aliases, addresses, forward – addresses and aliases for sendmail

SYNOPSIS

/etc/aliases
/etc/aliases.dir
/etc/aliases.pag
~/.forward

DESCRIPTION

These files contain mail addresses or aliases, recognized by **sendmail**(8), for the local host:

/etc/passwd	Mail addresses (usernames) of local users.
/etc/aliases	Aliases for the local host, in ASCII format. This file can be edited to add, update, or delete local mail aliases.
/etc/aliases.{dir,pag}	The aliasing information from **/etc/aliases**, in binary, **dbm**(3X) format for use by **sendmail**(8). The program **newaliases**(8), which is invoked automatically by **sendmail**(8), maintains these files.
~/.forward	Addresses to which a user's mail is forwarded (see **Automatic Forwarding**, below).

In addition, the Network Information Service (NIS) aliases map *mail.aliases* contains addresses and aliases available for use across the network.

Addresses

As distributed, **sendmail**(8) supports the following types of addresses:

Local Usernames

username

Each local *username* is listed in the local host's **/etc/passwd** file.

Local Filenames

pathname

Messages addressed to the absolute *pathname* of a file are appended to that file.

Commands

|*command*

If the first character of the address is a vertical bar, (|), **sendmail**(8) pipes the message to the standard input of the *command* the bar precedes.

TCP/IP-*standard Addresses*

username@domain

If *domain* does not contain any '.' (dots), then it is interpreted as the name of a host in the current domain. Otherwise, the message is passed to a *mailhost* that determines how to get to the specified domain. Domains are divided into subdomains separated by dots, with the top-level domain on the right. Top-level domains include:

.COM	Commercial organizations.
.EDU	Educational organizations.
.GOV	Government organizations.
.MIL	Military organizations.

For example, the full address of John Smith could be:

js@jsmachine.Podunk-U.EDU

if he uses the machine named **jsmachine** at Podunk University.

uucp(1C) *Addresses*

... [*host*!]*host*!*username*

These are sometimes mistakenly referred to as "Usenet" addresses. **uucp**(1C) provides links to numerous sites throughout the world for the remote copying of files.

Other site-specific forms of addressing can be added by customizing the **sendmail** configuration file. See the **sendmail**(8), and *System and Network Administration* for details. Standard addresses are recommended.

Aliases

Local Aliases

/etc/aliases is formatted as a series of lines of the form

aliasname: *address*[, *address*]

aliasname is the name of the alias or alias group, and *address* is the address of a recipient in the group. Aliases can be nested. That is, an *address* can be the name of another alias group. Because of the way **sendmail** performs mapping from upper-case to lower-case, an *address* that is the name of another alias group must not contain any upper-case letters.

Lines beginning with white space are treated as continuation lines for the preceding alias. Lines beginning with # are comments.

Special Aliases

An alias of the form:

owner–*aliasname*: *address*

directs error-messages resulting from mail to *aliasname* to *address*, instead of back to the person who sent the message.

An alias of the form:

aliasname: **:include:***pathname*

with colons as shown, adds the recipients listed in the file *pathname* to the *aliasname* alias. This allows a private list to be maintained separately from the aliases file.

NIS Domain Aliases

Normally, the aliases file on the master NIS server is used for the *mail.aliases* NIS map, which can be made available to every NIS client. Thus, the **/etc/aliases*** files on the various hosts in a network will one day be obsolete. Domain-wide aliases should ultimately be resolved into usernames on specific hosts. For example, if the following were in the domain-wide alias file:

jsmith:js@jsmachine

then any NIS client could just mail to **jsmith** and not have to remember the machine and username for John Smith. If an NIS alias does not resolve to an address with a specific host, then the name of the NIS domain is used. There should be an alias of the domain name for a host in this case. For example, the alias:

jsmith:root

sends mail on an NIS client to **root@podunk-u** if the name of the NIS domain is **podunk-u**.

Automatic Forwarding

When an alias (or address) is resolved to the name of a user on the local host, **sendmail** checks for a **.forward** file, owned by the intended recipient, in that user's home directory, and with universal read access. This file can contain one or more addresses or aliases as described above, each of which is sent a copy of the user's mail.

Care must be taken to avoid creating addressing loops in the **.forward** file. When forwarding mail between machines, be sure that the destination machine does not return the mail to the sender through the operation of any NIS aliases. Otherwise, copies of the message may "bounce". Usually, the solution is to change the NIS alias to direct mail to the proper destination.

A backslash before a username inhibits further aliasing. For instance, to invoke the **vacation**(1) program, user **js** creates a **.forward** file that contains the line:

\js, "|/usr/ucb/vacation js"

so that one copy of the message is sent to the user, and another is piped into the **vacation**(1) program.

FILES

/etc/passwd
/etc/aliases
~/.forward

SEE ALSO

uucp(1C), **vacation**(1), **dbm**(3X), **newaliases**(8), **sendmail**(8)

System and Network Administration

BUGS

Because of restrictions in **dbm**(3X) a single alias cannot contain more than about 1000 characters. Nested aliases can be used to circumvent this limit.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

ar – archive (library) file format

SYNOPSIS

#include <ar.h>

DESCRIPTION

The archive command **ar** combines several files into one. Archives are used mainly as libraries to be searched by the link-editor **ld**(1).

A file produced by **ar** has a magic string at the start, followed by the constituent files, each preceded by a file header. The magic number and header layout as described in the include file are:

```
#define ARMAG   "!<arch>\n"
#define SARMAG 8

#define ARFMAG "`\n"

struct ar_hdr {
        char    ar_name[16];
        char    ar_date[12];
        char    ar_uid[6];
        char    ar_gid[6];
        char    ar_mode[8];
        char    ar_size[10];
        char    ar_fmag[2];
};
```

The name is a blank-padded string. The **ar_fmag** field contains ARFMAG to help verify the presence of a header. The other fields are left-adjusted, blank-padded numbers. They are decimal except for **ar_mode**, which is octal. The date is the modification date of the file at the time of its insertion into the archive.

Each file begins on a even (0 mod 2) boundary; a NEWLINE is inserted between files if necessary. Nevertheless the size given reflects the actual size of the file exclusive of padding.

There is no provision for empty areas in an archive file.

The encoding of the header is portable across machines. If an archive contains printable files, the archive itself is printable.

Sun386i DESCRIPTION

The file produced by *ar* on Sun386i systems is identical to that described above with the following changes:

Each archive containing COFF files [see **coff**(5)] includes an archive symbol table. This symbol table is used by the link editor **ld** to determine which archive members must be loaded during the link edit process. The archive symbol table (if it exists) is always the first file in the archive (but is never listed) and is automatically created and/or updated by **ar**.

The *ar_name* field of the ar_hdr structure described above is blank-padded and slash (/) terminated. Common format archives can be moved from system to system as long as the portable archive command **ar** is used. Conversion tools such as **convert** exist to aid in the transportation of non-common format archives to this format.

Each archive file member begins on an even byte boundary; a NEWLINE is inserted between files if necessary. Nevertheless the size given reflects the actual size of the file exclusive of padding.

If the archive symbol table exists, the first file in the archive has a zero length name (i.e., **ar_name[0]** == '/'). The contents of this file are as follows:

- The number of symbols. Length: 4 bytes.
- The array of offsets into the archive file. Length: 4 bytes * "the number of symbols".

- The name string table. Length: *ar_size* – (4 bytes * ("the number of symbols" + 1)).

The number of symbols and the array of offsets are managed with *sgetl* and *sputl*. The string table contains exactly as many null terminated strings as there are elements in the offsets array. Each offset from the array is associated with the corresponding name from the string table (in order). The names in the string table are all the defined global symbols found in the common object files in the archive. Each offset is the location of the archive header for the associated symbol.

SEE ALSO

ar(1V), **ld**(1), **nm**(1)

Sun386i WARNINGS

strip(1) will remove all archive symbol entries from the header. The archive symbol entries must be restored via the **ts** option of the *ar*(1V) command before the archive can be used with the link editor *ld*(1).

BUGS

Filenames lose trailing blanks. Most software dealing with archives takes even an included blank as a name terminator.

NAME

audit.log – the security audit trail file

SYNOPSIS

```
#include <sys/label.h>
#include <sys/audit.h>
#include <sys/user.h>
```

DESCRIPTION

The **audit.log** file begins with a header record consisting of an **audit_header** structure followed by the previous audit file name. When the audit daemon is started (usually only at boot time), the previous audit file name is NULL.

```
struct audit_header {
        int     ah_magic;       /* magic number */
        time_t  ah_time;        /* the time */
        short   ah_namelen;     /* length of file name */
};
typedef struct audit_header audit_header_t;
```

The file may end with a trailer record consisting of an **audit_trailer** structure followed by the name of the next audit file.

```
struct audit_trailer {
        short   at_record_size;         /* size of this */
        short   at_record_type;         /* its type, a trailer */
        time_t  at_time;                /* the time */
        short   at_namelen;             /* length of file name */
};
typedef struct audit_trailer audit_trailer_t;
```

The **audit.log** file contains audit records in their raw form. The records are of varying size depending on the record type. Each record has a header which is an **audit_record** structure.

```
struct audit_record {
        short           au_record_size;         /* size of this */
        short           au_record_type;         /* its type */
        time_t          au_time;                /* the time */
        short           au_uid;                 /* real uid */
        short           au_auid;                /* audit uid */
        short           au_euid;                /* effective */
        short           au_gid;                 /* real group */
        short           au_pid;                 /* effective */
        int             au_errno;               /* error code */
        int             au_return;              /* a return value */
        blabel_t        au_label;               /* also ... */
        short           au_param_count;                 /* # of parameters */
};
typedef struct audit_record audit_record_t;
```

Immediately following the header is a set of two byte integers, the number of which exist for a given record is contained in the **au_param_count** field. These numbers are the lengths of the additional data items. The additional data items follow the list of lengths, the first length describing the first data item. Interpretation of this data is left to the program accessing it.

SEE ALSO

audit(2), **audit**(8)

Security Features Guide

NAME

audit_control – control information for system audit daemon

SYNOPSIS

/etc/security/audit/audit_control

DESCRIPTION

The **audit_control** file contains audit control information read by **auditd**(8). Each line consists of a title and a string, separated by a colon. There are no restrictions on the order of lines in the file, although some lines must appear only once. A line beginning with '#' is a comment.

Directory definition lines list the directories to be used when creating audit files, in the order in which they are to be used. The format of a directory line is:

dir: *directory-name*

where *directory-name* is the name of a directory in which to create audit files, with the form:

/etc/security/audit/*server***/***machine*

where *server* is the name of an audit file system on the machine where this audit directory resides, and *machine* is the name of the local machine, since audit files belonging to different machines are, by convention, stored in separate subdirectories of a single audit directory. The naming convention normally has *server* be the name of a server machine, and all clients mount **/etc/security/audit/***server* at the same location in their local file systems. If the same server exports several different file systems for auditing, their *server* names will, of course, be different.

The audit threshold line specifies the percentage of free space that must be present in the file system containing the current audit file. The format of the threshold line is:

minfree: *percentage*

where *percentage* is indicates the amount of free space required. If free space falls below this threshold, the audit daemon **auditd**(8) invokes the shell script **/etc/security/audit/audit_warn**. If no threshold is specified, the default is 0%.

The audit flags line specifies the default system audit value. This value is combined with the user audit value read from **/etc/security/passwd.adjunct** to form the process audit state. The user audit value overrides the system audit value. The format of a flags line is:

flags: *audit-flags*

where *audit-flags* specifies which event classes are to be audited. The character string representation of *audit-flags* contains a series of flag names, each one identifying a single audit class, separated by commas. A name preceded by '–' means that the class should be audited for failure only; successful attempts are not audited. A name preceded by '+' means that the class should be audited for success only; failing attempts are not audited. Without a prefix, the name indicates that the class is to be audited for both successes and failures. The special string **all** indicates that all events should be audited; **–all** indicates that all failed attempts are to be audited, and **+all** all successful attempts. The prefixes ^, ^–, and ^+ turn off flags specified earlier in the string (^– and ^+ for failing and successful attempts, ^ for both). They are typically used to reset flags.

The following table lists the audit classes:

short name	long name	short description
dr	**data_read**	Read of data, open for reading, etc.
dw	**data_write**	Write or modification of data
dc	**data_create**	Creation or deletion of any object
da	**data_access_change**	Change in object access (modes, owner)
lo	**login_logout**	Login, logout, creation by **at**(1)
ad	**administrative**	Normal administrative operation
p0	**minor_privilege**	Privileged operation
p1	**major_privilege**	Unusual privileged operation

EXAMPLE

Here is a sample **/etc/security/audit_control** file for the machine eggplant:

```
dir: /etc/security/audit/jedgar/eggplant
dir: /etc/security/audit/jedgar.aux/eggplant
#
# Last-ditch audit file system when jedgar fills up.
#
dir: /etc/security/audit/global/eggplant
minfree: 20
flags: lo,p0,p1,ad,-all,^-da
```

This identifies server **jedgar** with two file systems normally used for audit data, another server **global** used only when **jedgar** fills up or breaks, and specifies that the warning script is run when the file systems are 80% filled. It also specifies that all logins, privileged and administrative operations are to be audited (whether or not they succeed), and that failures of all types except failures to access data are to be audited.

FILES

/etc/security/audit/audit_control
/etc/security/audit/audit_warn
/etc/security/audit/*/*/*
/etc/security/passwd_adjunct

SEE ALSO

at(1), **audit**(2), **getfauditflags**(3), **audit.log**(5), **audit**(8), **auditd**(8)

NAME

audit_data – current information on audit daemon

SYNOPSIS

/etc/security/audit/audit_data

DESCRIPTION

The **audit_data** file contains information about the audit daemon. The file contains the process ID of the audit daemon, and the pathname of the current audit log file. The format of the file is:

<pid>:<pathname>

Where *pid* is the process ID for the audit daemon, and *pathname* is the full pathname for the current audit log file.

EXAMPLE

64:/etc/security/audit/auditserv/auditclient/2df0504

FILES

/etc/security/audit/audit_data

SEE ALSO

audit(2), **audit.log**(5), **audit**(8), **auditd**(8)

NAME

auto.home – automount map for home directories

SYNOPSIS

/etc/auto.home

AVAILABILTITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

auto.home resides in the **/etc** directory, and contains **automount**(8) map entries for user's home directories. On Sun386i systems, this file is used to build the **auto.home** Network Information Service (NIS) map used by **automount** at system startup and reads the **auto.master** NIS database, which contains an entry for **auto.home** and **/home** . The **auto.home** map contains entries for each username in the NIS **passwd** map, and the *hostname*:*/directory* to NFS mount.

References to **/home/***username* are translated by the automount daemon using the **auto.home** map, and the directory specified in the map entry is **nfs** mounted and that directory returned to the user's program.

User accounts created using **snap**(1) or **logintool**(8) have **passwd**(5) entries where the initial (home) directory name is, in the form **/home/***username*. **snap** and **logintool** also automatically create the **auto.home** entry for a user account. The format of the entry is described in **automount**(8). An example entry is:

mtravis system2:/export/home/users/mtravis

Thus, when the user **mtravis** logs into a Sun386i systems, the automounter automatically mounts his home directory from **system2**. This allows a user to log in to any Sun386i workstation on the network and be automatically placed in their home directory.

The convention for the format of home directory names used by **snap** and **logintool** is:

/export/home/*groupname/username*

Note: this is a different map and mechanism for home directories than the one that the automount daemon provides with the **–homes** switch. This is because the Sun386i convention for the format of home directory names differs and provides directories that can be used as mount points on a per user and per group basis.

FILES

/etc/auto.home

SEE ALSO

snap(1), **passwd**(5), **automount**(8), **logintool**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

auto.vol – automount map for volumes

SYNOPSIS

/etc/auto.vol

AVAILABILTITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

auto.vol resides in the **/etc** directory, and contains **automount**(8) map entries for **volumes**. On Sun386i systems, this file is used to build the **auto.vol** Network Information Service (NIS) map used by **automount**(8) at system startup. **automouunt** reads the **auto.master** NIS map, which contains an entry for **auto.vol** and **/vol**.

References to **/vol/***volume_name* are translated by the automount daemon using the **auto.vol** map, and the directory specified in the map entry is mounted.

The concept of a volume is that it is a self contained directory hierarchy that can be NFS mounted. It is referenced using a known *volume_name*. The use of an automount map is suggested so that the volume and its contents can be referenced through **/vol**. This is advantageous because location-transparency (that is, which host the volume is on) and replication of read-only volumes can be provided using the automount mechanism. The format of the entry is described in **automount**(8). An example entry is:

archive system4:/export/archive

In the above example, the **archive** volume is currently on line on **system4**. Users and programs can reference it via **/vol/archive**. If for some reason the volume had to be moved to another system, **system2** for example, the network or system administrator simply edits the map entry for the archive volume and changes the hostname to **system2** and then rebuilds the NIS maps.

archive system2:/export/archive

Users and programs can continue to refer to the archive volume using **/vol/archive**, unaware that the volume was moved to another system.

FILES

/etc/auto.vol

SEE ALSO

automount(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

bar – tape archive file format

DESCRIPTION

bar(1), (the tape archive command) dumps several files into one, in a medium suitable for transportation. This format is not compatible with the format generated by **tar**(1).

A *bar tape* or file is a series of blocks. Each block is of size **TBLOCK.** A file on the tape is represented by a header block that describes the file, followed by zero or more blocks that give the contents of the file. At the end of the tape are two blocks filled with binary zeros, as an EOF indicator.

The blocks are grouped for physical I/O operations. Each group of *n* blocks (where *n* is set by the **b** keyletter on the **bar**(1) command line — default is 20 blocks) is written with a single system call; on nine-track tapes, the result of this write is a single tape record. The last group is always written at the full size, so blocks after the two zero blocks contain random data. On reading, the specified or default group size is used for the first read, but if that read returns less than a full tape block, the reduced block size is used for further reads, unless the **B** keyletter is used.

The header block looks like:

```
#define TBLOCK512

union hblock {
        char dummy[TBLOCK];
        struct header {
                char mode[8];
                char uid[8];
                char gid[8];
                char size[12];
                char mtime[12];
                char chksum[8];
                char rdev[8];
                char linkflag;
                char bar_magic[2];
                char volume_num[4];
                char compressed;
                char date[12];
                char start_of_name;
        } dbuf;
};
```

start_of_name is a null-terminated string. *date* is the date of the archive. *bar_magic* is a special number indicating that this is a **bar** archive. *rdev* is the device type, for files that are devices. The other fields are zero-filled octal numbers in ASCII. Each field (of width w) contains w-2 digits, a space, and a null, except *size*, *rdev*, and *mtime*, which do not contain the trailing null. *start_of_name* is the name of the file, as specified on the *bar* command line. Files dumped because they were in a directory that was named in the command line have the directory name as prefix and */filename* as suffix. *mode* is the file mode, with the top bit masked off. *uid* and *gid* are the user and group numbers that own the file. *size* is the size of the file in bytes. Links and symbolic links, and special files, are dumped with this field specified as zero. *mtime* is the modification time of the file at the time it was dumped. *chksum* is a decimal ASCII value that represents the sum of all the bytes in the header block. When calculating the checksum, the *chksum* field is treated as if it were all blanks. *linkflag* is ASCII 0 if the file is "normal" or a special file, 1 if it is an hard link, 2 if it is a symbolic link, and 3 if it is a special file (device or FIFO). The name linked-to, if any, is in a null-terminated string, following *start_of_name*. Unused fields of the header are binary zeros (and are included in the checksum).

The first time a given i-node number is dumped, it is dumped as a regular file. The second and subsequent times, it is dumped as a link instead. Upon retrieval, if a link entry is retrieved, but not the file it was linked to, an error message is printed and the tape must be manually re-scanned to retrieve the linked-to file.

When the **H** modifier is used with **bar** , an additional header block (one that does not pertain to a particular file) is written to the first block of each volume of the archive. The header ID, as specified on the command line, is copied to *start_of_name*. The size reflects the number of bytes to skip to the start of the first full file (always zero on the first volume).

The encoding of the header is designed to be portable across machines.

SEE ALSO

bar(1)

NAME

boards.pc – information about AT- and XT-compatible boards for DOS windows

SYNOPSIS

/etc/dos/defaults/boards.pc

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **boards.pc** file stores information about AT- and XT-compatible boards installed on a system.

Only the super-user may alter the file.

The file format is as follows, with entries separated by SPACE or TAB characters:

Board-name *I/O port range* *IRQ* *DMA* *Memory* *Options*

Board-name

The name of the board as it will appear in the DOS Windows Device menu. Use any name that is not longer than 19 characters.

I/O port range

Most boards have I/O addresses through which they exchange information with the workstation. For boards that will be used by DOS, the I/O address is entered in the **boards.pc** file, directly to the right of the board name.

Certain I/O addresses are already used by DOS Windows emulated devices (such as drive C and the DOS printers), and by built-in system hardware. The following list shows the AT-bus I/O address spaces:

Address	DOS Use
1F8-1FF *	**Hard disk (C:) emulation**
218-21F	**Expanded memory**
230-23F	**Bus mouse emulation**
278-27F	**Parallel port 2 (usually accessed through LPT3)**
378-37F *	**Parallel port 1 (usually accessed through LPT2)**
3B0-3BF	**Monochrome display adapter**
3D0-3DF	**Color display adapter**
3F0-3F7 *	**Diskette controller**

An address marked with an asterisk cannot be replaced by a board. When the board you are installing uses one of these addresses, or it uses the same address as another board that is already installed, change the jumpers or switch settings on your board to use a different address. If you add a board that occupies one of these address spaces, DOS ignores the entry. An address not marked with an asterisk may be used for a board you are installing, as long as you do not plan to use the emulated device at that address.

Adding an I/O Address Entry to boards.pc:

If the board uses addresses that can be contained within one eight-address block, note the block base address and include it in the *I/O port range* column of the **boards.pc** file. When using a multiple-block address, specify the base address of each block. For example, when entering a two-block address, specify the base addresses of both the first and second blocks, and separated with a SPACE character. Suppose you have a board with a two-block I/O address space that begins at 380. You would specify 380 388 in the **boards.pc** file's *I/O port range* column.

IRQ Some boards send periodic signals asking DOS to delay whatever it is doing and accept information from the device. These signals are known as **interrupt requests**, or more simply, as **interrupts**. The following chart shows the interrupt levels available under DOS Windows. Valid interrupt levels are 1 to 15, although some of these are reserved for emulated DOS devices.

Interrupt Level	Availability
0	**Unavailable; used for timer emulation**
1	**Unavailable; used for keyboard emulation**
2	**Unavailable; used for interrupt controller 2 cascade**
3	**Available for board, unless COM2 emulation in use (specified in setup.pc)**
4	**Available for board, unless COM1 emulation in use (specified in setup.pc)**
5	**Available for board, unless LPT3 emulation in use (specified in setup.pc)**
6	**Unavailable; used for diskette drive emulation**
7	**Unavailable; used by built-in parallel port**
8	**Unavailable; used for real-time clock emulation**
9	**Available for board**
10	**Available for board**
11	**Available for board**
12	**Available for board**
13	**Unavailable; used for 8087 numeric coprocessor emulation**
14	**Unavailable; used for hard disk emulation**
15	**Available for board**

To ensure that signals do not become confused, set each board or emulated device that uses interrupts for a different interrupt level. Normally, interrupt settings are changed by pressing small switches or moving metal jumpers on the board itself. Consult the manual of the board you are installing for details on how this is done. In addition to the changes required on the board itself, make sure that the interrupt level in your **boards.pc** file matches the setting on the card. For example, if a board's physical interrupt was previously 3, and you change it to 4 by altering switch settings or board jumpers, make a corresponding change in the **boards.pc** file. If the card uses a DOS driver, you may also need to make changes in **C:CONFIG.SYS** or other files to reflect the switch settings on the card.

Adding an Interrupt Entry to boards.pc

Some boards do not generate interrupts, and therefore will not have an interrupt level listed in their manuals. If this is the case, leave the *IRQ* column empty. For boards where an interrupt level is required, enter the letters **irq** followed by the appropriate number in the **boards.pc** file, as shown in EXAMPLES below.

DMA Certain boards use direct memory access (DMA) channels to ensure speedy transfer of large quantities of data. DMA channels 0, 1, 3, and 5 are available. Each DOS or SunOS DMA board on the system must be assigned a unique DMA channel. When two or more boards expect to use DMA channel 1, physically alter DMA settings on one of the boards so that it uses a different channel (such as DMA channel 3). Normally these settings are changed by pressing small switches or moving metal jumpers on the board itself. Consult the manual for the board you are installing for details on changing a DMA channel setting.

Adding a DMA Entry to boards.pc

When the board you are installing uses a DMA channel, include a **dma** entry for that board. For example, when the board is set up to use DMA channel 3, the entry can look like this:

MYBOARD 200 208 irq 2 dma 3

Memory

Some boards are equipped with memory chips for DOS. Because this memory is "mapped" (transferred) into DOS memory so that DOS can read it, the boards are called *memory mapped boards*. When you install such a board, include a **mem** entry with the following format:

mem *address size*

The *address* is the starting address of the memory segment, in hexadecimal notation. Enter the size of the memory block in kilobytes, in decimal notation. The following example is for a board that starts mapped memory at the address $DE00 and uses a block of 8 kilobytes.

MYBOARD 258 irq 5 dma 3 mem de00 8

When determining the size of the memory block, be careful not to confuse DOS address size (the number you should use) with actual on-board memory (the number you should not use). For example, a LIM memory board might have 2 megabytes of on-board memory, yet may require only 64 kilobytes of DOS address space for its memory mapping. Therefore, the number to use for the **mem** entry is 64.

Options

reboot

Certain boards require DOS rebooting before they work. These same boards require that you reboot DOS after you have finished using them. You can set up DOS to reboot the current DOS window automatically whenever the board is attached. DOS displays a confirmatory alert before rebooting.

To force DOS to reboot when you attach the board, add the word **reboot** at the end of the **boards.pc** line for that board, as shown in the following example:

MYBOARD 3e8 mem a000 192 reboot

If you choose to omit the **reboot** instruction, you can enable the board by attaching it and then manually rebooting:

1. Choose **Attached from the Device** menu to enable the board.
2. Choose **Reboot DOS Window.**

To detach such a board from a DOS window, choose **Detach** and then reboot the DOS window.

shared

You can specify that a device is to be shared between windows, rather than being reserved for use by one window at a time. Generally, you should do this only with devices, such as joysticks, which can fluidly move from one DOS window to another. To designate a device as shared, place the word **shared** at the very end of the **boards.pc** line:

Joystick 200 shared

Determining Board Information

In many cases, you may need to determine whether a board you are installing will conflict with other devices on the system. Also, you sometimes may need to install a board for which there is no entry in the **boards.pc** file. In most cases, the instruction manual included with the board you are installing should contain the technical information you need, including:

The I/O port addresses at which the board is accessed. One or more blocks can be reserved, and there are eight consecutive addresses per block.

The board's interrupt level, if the board generates interrupts.

The DMA channel number, if the board uses a direct memory access channel.

Memory mapping information, if the board maps data into DOS memory.

If the board's manual does not provide such information, contact the manufacturer.

EXAMPLES

The following is an example of a **boards.pc** file:

```
#COM2             2f8                              irq 3
#Joystick         200                                                    shared
#EGA              3b0 3b8 3c0 3c8 3d0 3d8                  mem a000 192  reboot
#VGA              3b0 3b8 3c0 3c8 3d0 3d8 102 2e8          mem a000 192
#3COM-3C501       300 308                          irq 3  dma 1
#TOPS-FlashTalk   398                              irq 3
#IBM-3363-Worm    258                              irq 5  dma 3  mem de00 8   reboot
#Plus-Hardcard20  320                              irq 5  dma 3  mem ca00 8   reboot
#HP-Basic         390                              irq 3
#DCA-IRMA1        220 228
#DCA-IRMA2        220 228 280 288
#Bernoulli-A220H  350                                                   reboot
#WD8003E          280 288 290 298                  irq 5       mem d000 8
#NI5210           360                              irq 5       mem c000 16
#NIC                                               irq 5       mem d000 32
#LPT2             278                              irq 5
```

FILES

/usr/lib/help/*/*

SEE ALSO

dos(1), **setup.pc**(5)

Sun386i Advanced Skills

NAME

bootparams – boot parameter data base

SYNOPSIS

/etc/bootparams

DESCRIPTION

The **bootparams** file contains the list of client entries that diskless clients use for booting. For each diskless client the entry should contain the following information:

name of client
a list of keys, names of servers, and pathnames.

The first item of each entry is the name of the diskless client. The subsequent item is a list of keys, names of servers, and pathnames.

Items are separated by TAB characters.

A client entry in the local **/etc/bootparams** file supersedes an entry in the corresponding Network Information Service (NIS) map.

EXAMPLE

Here is an example of the **/etc/bootparams** taken from a SunOS system.

```
myclient        root=myserver:/nfsroot/myclient \
        swap=myserver:/nfsswap/myclient \
        dump=myserver:/nfsdump/myclient
```

FILES

/etc/bootparams

SEE ALSO

bootparamd(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

bootservers – NIS bootservers file

SYNOPSIS

/etc/bootservers

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0*.x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **bootservers** file is an ASCII file that resides in the **/etc** directory on the Network Information Service (NIS) master server. The file contains basic information about each host providing boot services for clients on the network. This file contains a one-line entry for each boot server, where each field *must* be separated by a TAB character:

system type client_limit swap_size tmp_size root_minfree swap_minfree

The entries in the file have the following descriptions:

system	is the name of a boot server. This field contains only lowercase and numeric characters, must start with a lower-case character, and must not be longer than 32 characters.
type	Currently, the only legal value is 3.
client_limit	indicates the maximum number of diskless clients the server is willing to accept.
swap_size	default swap size per client (in kilobytes).
tmp_size	default tmp size per client (in kilobytes).
root_minfree	minimum amount of disk space in the server's client-root partition after a client is added (in kilobytes).
swap_minfree	minimum amount of disk space in the server's client-swap partition after a client is added (in kilobytes).

EXAMPLE

Here is a sample **bootservers** file entry:

```
polaris 3 2 16000 8000 40000 0
```

FILES

/etc/bootservers

SEE ALSO

System and Network Administration,
Sun386i Advanced Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

coff – common assembler and link editor output

SYNOPSIS

```
#include <filehdr.h>
#include <aouthdr.h>
#include <scnhdr.h>
#include <reloc.h>
#include <linenum.h>
#include <storclass.h>
#include <syms.h>
```

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The output from the link editor and the assembler (named **a.out** by default) is in COFF format (Common Object File Format) on the Sun386i system.

A common object file consists of a file header, a system header (if the file is link editor output), a table of section headers, a data section, relocation information, (optional) line numbers, a symbol table, and a string table. The general format looks like this:

file-header
system-header
section-headers
data
relocation
line-numbers
symbol-table
string-table

section-headers contains a number of section headers:

section 1 header
...
section n header

Similarly, *data*, *relocation*, and *line-numbers* are each divided into *n* sections.

The last three parts of an object file (line numbers, symbol table and string table) may be missing if the program was linked with the **–s** option of **ld**(1) or if they were removed by **strip**(1). Also note that the relocation information will be absent after linking unless the **–r** option of **ld**(1) was used. The string table exists only if the symbol table contains symbols with names longer than eight characters.

The sizes of each section (contained in the header, discussed below) are in bytes.

When an **a.out** file is loaded into memory for execution, three logical segments are set up: the text segment, the data segment (initialized data followed by uninitialized, the latter actually being initialized to all 0's), and a stack. The text segment starts at location 0x1000 by default.

The **a.out** file produced by **ld**(1) has the magic number 0413 in the first field of the system header. The headers (file header, system header, and section headers) are loaded at the beginning of the text segment and the text immediately follows the headers in the user address space. The first text address will equal 0x1000 plus the size of the headers, and will vary depending upon the number of section headers in the **a.out** file. In an **a.out** file with three sections (.text, .data, .bss, and .comment), the first text address is at 0x000010D0. The text segment is not writable by the program; if other processes are executing the same **a.out** file, the processes will share a single text segment.

The data segment starts at the next 4K boundary past the last text address. The first data address is determined by the following: If an **a.out** file were split into 4K chunks, one of the chunks would contain both the end of text and the beginning of data. When the **a.out** file is loaded into memory for execution, that chunk will appear twice; once at the end of text and once at the beginning of data (with some unused space in between). The duplicated chunk of text that appears at the beginning of data is never executed; it is duplicated so that the operating system may bring in pieces of the file in multiples of the page size without having to realign the beginning of the data section to a page boundary. Therefore the first data address is the sum of the next segment boundary past the end of text plus the remainder of the last text address divided by 4K. If the last text address is a multiple of 4K no duplication is necessary.

On the Sun386i computer the stack begins at location 0xFBFFFFFF and grows toward lower addresses. The stack is automatically extended as required. The data segment is extended only as requested by the **brk**(2) system call.

For relocatable files the value of a word in the text or data portions that is not a reference to an undefined external symbol is exactly the value that will appear in memory when the file is executed. If a word in the text involves a reference to an undefined external symbol, there will be a relocation entry for the word, the storage class of the symbol-table entry for the symbol will be marked as an "external symbol", and the value and section number of the symbol-table entry will be undefined. When the file is processed by the link editor and the external symbol becomes defined, the value of the symbol will be added to the word in the file.

File Header

The format of the file header is:

```
struct filehdr
{
        unsigned short  f_magic;  /* magic number */
        unsigned short  f_nscns;  /* number of sections */
        long            f_timdat; /* time and date stamp */
        long            f_symptr; /* file ptr to symtab */
        long            f_nsyms;  /* # symtab entries */
        unsigned short  f_opthdr; /* sizeof(opt hdr) */
        unsigned short  f_flags;  /* flags */
};
```

System Header

The format of the system header is:

```
typedef struct aouthdr
{
        short   magic;          /* magic number */
        short   vstamp;         /* version stamp */
        long    tsize;          /* text size in bytes, padded */
        long    dsize;          /* initialized data (.data) */
        long    bsize;          /* uninitialized data (.bss) */
        long    entry;          /* entry point */
        long    text_start;     /* base of text used for this file */
        long    data_start;     /* base of data used for this file */
} AOUTHDR;
```

Section Header

The format of the section header is:

```
struct scnhdr
{
        char            s_name[SYMNMLEN];/* section name */
        long            s_paddr;  /* physical address */
        long            s_vaddr;  /* virtual address */
        long            s_size;   /* section size */
        long            s_scnptr; /* file ptr to raw data */
        long            s_relptr; /* file ptr to relocation */
        long            s_lnnoptr;/* file ptr to line numbers */
        unsigned shorts_nreloc;  /* # reloc entries */
        unsigned shorts_nlnno;   /* # line number entries */
        long            s_flags;  /* flags */
};
```

Relocation

Object files have one relocation entry for each relocatable reference in the text or data. If relocation information is present, it will be in the following format:

```
struct reloc
{
        long        r_vaddr;  /* (virtual) address of reference */
        long        r_symndx;/* index into symbol table */
        ushort      r_type;   /* relocation type */
};
```

The start of the relocation information is **s_relptr** from the section header. If there is no relocation information, **s_relptr** is 0.

Line Number

The **cc**(1V) command generates an entry in the object file for each C source line on which a breakpoint is possible (when invoked with the **–g** option. Users can refer to line numbers when using the appropriate debugger, such as **dbx**(1)). The structure of these line number entries appears below.

```
struct lineno
{
        union
        {
                long    l_symndx ;
                long    l_paddr ;
        }               l_addr ;
        unsigned shortl_lnno ;
} ;
```

Numbering starts with one at the top of the source file and increments independent of transition between functions. The initial line number entry for a function has **l_lnno** equal to zero, and the symbol table index of the function's entry is in **l_symndx**. Otherwise, **l_lnno** is non-zero, and **l_paddr** is the physical address of the code for the referenced line. Thus the overall structure is the following:

l_addr	**l_lnno**
function symtab index	0
physical address	line
physical address	line
...	
function symtab index	0

```
physical address        line
physical address        line
...
```

Symbol Table

The format of each symbol in the symbol table is described by the syment structure, shown below. This structure is compatible with System V COFF, but has an added **_n_dbx** structure which is needed by **dbx**(1).

```
#define SYMNMLEN 8
#define FILNMLEN 14
#define DIMNUM   4

struct syment
{
  union                         /* all ways to get a symbol name */
  {
    char            _n_name[SYMNMLEN]; /* name of symbol */
    struct
    {
      long          _n_zeroes;    /* == 0L if in string table */
      long          _n_offset;    /* location in string table */
    } _n_n;
    char            *_n_nptr[2]; /* allows overlaying */
    struct
    {
      char          _n_leading_zero; /* null char */
      char          _n_dbx_type; /* stab type */
      short         _n_dbx_desc; /* value of desc field */
      long          _n_stab_ptr; /* table ptr */
    } _n_dbx;
  } _n;
  long              n_value;      /* value of symbol */
  short             n_scnum;      /* section number */
  unsigned short    n_type;       /* type and derived type */
  char              n_sclass;     /* storage class */
  char              n_numaux;     /* number of aux entries */
};

#define n_name      _n._n_name
#define n_zeroes    _n._n_n._n_zeroes
#define n_offset    _n._n_n._n_offset
#define n_nptr      _n._n_nptr[1]
```

The storage class member (**n_sclass**) is set to one of the constants defined in **<storclass.h>**. Some symbols require more information than a single entry; they are followed by *auxiliary entries* that are the same size as a symbol entry. The format follows:

```
union auxent {
        struct {
                long    x_tagndx;
                union {
                        struct {
                                unsigned short  x_lnno;
                                unsigned short  x_size;
                        } x_lnsz;
                        long    x_fsize;
                } x_misc;
                union {
                        struct {
                                long    x_lnnoptr;
                                long    x_endndx;
                        }  x_fcn;
                        struct {
                                unsigned short  x_dimen[DIMNUM];
                        } x_ary;
                } x_fcnary;
                unsigned short  x_tvndx;
        } x_sym;

        struct {
                char    x_fname[FILNMLEN];
        } x_file;

        struct {
                long        x_scnlen;
                unsigned short  x_nreloc;
                unsigned short  x_nlinno;
        } x_scn;

        struct {
                long            x_tvfill;
                unsigned short  x_tvlen;
                unsigned short  x_tvran[2];
        } x_tv;
};
```

Indexes of symbol table entries begin at *zero*. The start of the symbol table is **f_symptr** (from the file header) bytes from the beginning of the file. If the symbol table is stripped, **f_symptr** is 0. The string table (if one exists) begins at **f_symptr** + (**f_nsyms** * SYMESZ) bytes from the beginning of the file.

SEE ALSO

as(1), **cc**(1V), **ld**(1), **brk**(2), **ldfcn**(3)

NAME

core – format of memory image file

SYNOPSIS

#include <sys/core.h>

DESCRIPTION

The operating system writes out a memory image of a terminated process when any of various errors occur. See **sigvec**(2) for the list of reasons; the most common are memory violations, illegal instructions, bus errors, and user-generated quit signals. The memory image is called **core** and is written in the process's working directory (provided it can be; normal access controls apply). Set-user-ID and set-group-ID programs do not produce core files when they terminate as this would cause a security loophole.

The maximum size of a **core** file is limited by **setrlimit** (see **getrlimit**(2)). Files which would be larger than the limit are not created.

The core file consists of a **core** structure, as defined in the **<sys/core.h>** file, followed by the data pages and then the stack pages of the process image. The **core** structure includes the program's header, the size of the text, data, and stack segments, the name of the program and the number of the signal that terminated the process. The program's header is described by the **exec** structure defined in the **<sys/exec.h>** file, except on Sun386i systems.

```
struct core {
        int     c_magic;        /* Corefile magic number */
        int     c_len;          /* Sizeof (struct core) */
        struct  regs c_regs;    /* General purpose registers */
        struct  exec c_aouthdr; /* A.out header */
        int     c_signo;        /* Killing signal, if any */
        int     c_tsize;        /* Text size (bytes) */
        int     c_dsize;        /* Data size (bytes) */
        int     c_ssize;        /* Stack size (bytes) */
        char    c_cmdname[CORE_NAMELEN + 1]; /* Command name */
        struct  fpu c_fpu;      /* external FPU state */
        int     c_ucode;        /* Exception no. from u_code */
};
```

The members of the structure are:

c_magic The magic number CORE_MAGIC, as defined in **<sys/core.h>**.

c_len The length of the **core** structure in the core file. This need not be equal to the current size of a **core** structure as defined in **<sys/core.h>**, as the core file may have been produced on a different release of the SunOS operating system.

c_regs The general purpose registers at the time the core file was produced. This structure is machine-dependent.

c_aouthdr The executable image header of the program.

c_signo The number of the signal that terminated the process; see **sigvec**(2).

c_tsize The size of the text segment of the process at the time the core file was produced.

c_dsize The size of the data segment of the process at the time the core file was produced. This gives the amount of data space image in the core file.

c_ssize The size of the stack segment of the process at the time the core file was produced. This gives the amount of stack space image in the core file.

c_cmdname The first CORE_NAMELEN characters of the last component of the path name of the program.

c_fpu The status of the floating point hardware at the time the core file was produced.

c_ucode The signal code of the signal that terminated the process, if any. See **sigvec**(2).

SEE ALSO

adb(1), **dbx**(1), **getrlimit**(2), **sigvec**(2)

NAME

cpio – format of cpio archive

DESCRIPTION

The old format *header* structure, when the **–c** option of **cpio** is not used, is:

```
struct {
        short   h_magic,
                h_dev;
        ushort  h_ino,
                h_mode,
                h_uid,
                h_gid;
        short   h_nlink,
                h_rdev,
                h_mtime[2],
                h_namesize,
                h_filesize[2];
        char    h_name[h_namesize rounded to a word];
} Hdr;
```

The byte order here is that of the machine on which the tape was written. If the tape is being read on a machine with a different byte order, you have to use **swab**(3) after reading the header. You can determine what byte order the tape was written with by examining the *h_magic* field; if it is equal to 0143561 (octal), which is the standard magic number 070707 (octal) with the bytes swapped, the tape was written in a byte order opposite to that of the machine on which it is being read. If you are producing a tape to be read on a machine with the opposite byte order to that of the machine on which it is being produced, you can use **swap** before writing the header.

When the **–c** option is used, the *header* information is described by the statement below:

```
sscanf(Chdr, "%6o%6o%6o%6o%6o%6o%6o%6o%11lo%6o%11lo%s",
        &Hdr.h_magic, &Hdr.h_dev, &Hdr.h_ino, &Hdr.h_mode,
        &Hdr.h_uid, &Hdr.h_gid, &Hdr.h_nlink, &Hdr.h_rdev,
        &Hdr.h_mtime, &Hdr.h_namesize, &Hdr.h_filesize, &Hdr.h_name);
```

Longtime and *Longfile* are equivalent to *Hdr.h_mtime* and *Hdr.h_filesize*, respectively. The contents of each file is recorded in an element of the array of varying length structures, *archive*, together with other items describing the file. Every instance of *h_magic* contains the constant 070707 (octal). The items *h_dev* through *h_mtime* have meanings explained in **stat**(2V). The length of the null-terminated path name *h_name*, including the null byte, is given by *h_namesize*.

The last record of the *archive* always contains the name **TRAILER!!!**. Special files, directories, and the trailer, are recorded with *h_filesize* equal to zero. Symbolic links are recorded similarly to regular files, with the "contents" of the file being the name of the file the symbolic link points to.

SEE ALSO

cpio(1), **find**(1), **stat**(2V), **swab**(3)

NAME

crontab – table of times to run periodic jobs

SYNOPSIS

/var/spool/cron/crontabs/*

DESCRIPTION

The **cron** utility is a permanent process, started by **/etc/rc.local**. **cron** consults the files in the directory **/var/spool/cron/crontabs** to find out what tasks are to be done, and at what time.

Each line in a **crontab** file consists of six fields, separated by spaces or tabs, as follows:

minutes hours day-of-month month day-of-week command

minutes	Minutes field, which can have values in the range 0 through 59.
hours	Hours field, which can have values in the range 0 through 23.
day-of-month	Day of the month, in the range 1 through 31.
month	Month of the year, in the range 1 through 12.
day-of-week	Day of the week, in the range 0 through 6. Sunday is day 0 in this scheme of things. For backward compatibility with older systems, Sunday may also be specified as day 7.
command	Command to be run. A percent character in this field (unless escaped by \) is translated to a NEWLINE character. Only the first line (up to a % or end of line) of the command field is executed by the Shell. The other lines are made available to the command as standard input.

Any of fields 1 through 5 can be a list of values separated by commas. A value can either be a number, or a pair of numbers separated by a hyphen, indicating that the job is to be done for all the times in the specified range. If a field is an asterisk character (*) it means that the job is done for all possible values of the field.

Note: the specification of days may be made by two fields (day of the month and day of the week). If both are specified as a list of elements, both are adhered to. For example,

0 0 1,15 * 1

would run a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to *. For example,

0 0 * * 1

would run a command only on Mondays.

The command is run from your home directory with an **arg0** of **sh**. Users who desire to have their **.profile** executed must explicitly do so in the command. **cron** supplies a default environment for every shell, defining **HOME, LOGNAME, USER, SHELL(=/bin/sh)**, and **PATH(=:/usr/ucb:/bin:/usr/bin)**.

NOTE: Users should remember to redirect the standard output and standard error of their commands! If this is not done, any generated output or errors will be mailed to the user.

Lines that start with # are treated as comments.

EXAMPLES

```
0 0 * * * calendar –
15 0 * * * /usr/etc/sa –s >/dev/null
15 4 * * * find /var/preserve –mtime +7 –a –exec rm –f {} ;
40 4 * * * find / –name '#*' –atime +3 –exec rm –f {} ;
0 0 * * 1-5 /usr/local/weekdays
0 0 * * 0,6 /usr/local/weekends
```

The **calendar** command runs at minute 0 of hour 0 (midnight) of every day. The **/usr/etc/sa** command runs at 15 minutes after midnight every day. The two **find** commands run at 15 minutes past four and at 40 minutes past four, respectively, every day of the year. The **/usr/local/weekdays** command is run at midnight on weekdays. Finally, the **/usr/local/weekends** command is run at midnight on weekends.

FILES

/var/spool/cron/crontabs/* tables of times to run periodic jobs

/etc/rc.local

.profile

SEE ALSO

cron(8), **rc**(8)

NAME

dir – format of directories

SYNOPSIS

```
#include <sys/types.h>
#include <sys/dir.h>
```

DESCRIPTION

A directory behaves exactly like an ordinary file, save that no user may write into a directory and directories must be read using the **getdirentries**(2) system call or the **directory**(3V) library routines. The fact that a file is a directory is indicated by a bit in the flag word of its inode entry; see **fs**(5).

A directory consists of some number of blocks of DIRBLKSIZ bytes, where DIRBLKSIZ is chosen such that it can be transferred to disk in a single atomic operation (512 bytes on most machines):

```
#ifdef KERNEL
#define DIRBLKSIZ DEV_BSIZE
#else
#define DIRBLKSIZ 512
#endif

#define MAXNAMLEN 255
```

Each DIRBLKSIZ byte block contains some number of directory entry structures, which are of variable length. Each directory entry has a **struct direct** at the front of it, containing its inode number, the length of the entry, and the length of the name contained in the entry. These are followed by the name padded to a 4-byte boundary with null bytes. All names are guaranteed null-terminated. The maximum length of a name in a directory is MAXNAMLEN.

The macro **DIRSIZ(dp)** gives the amount of space required to represent a directory entry. Free space in a directory is represented by entries that have:

```
dp->d_reclen > DIRSIZ(dp)
```

All DIRBLKSIZ bytes in a directory block are claimed by the directory entries. This usually results in the last entry in a directory having a large **dp->d_reclen**. When entries are deleted from a directory, the space is returned to the previous entry in the same directory block by increasing its **dp->d_reclen**. If the first entry of a directory block is free, then its **dp->d_ino** is set to 0. Entries other than the first in a directory do not normally have **dp->d_ino** set to 0.

The **DIRSIZ** macro gives the minimum record length which will hold the directory entry. This requires the amount of space in struct direct without the **d_name** field, plus enough space for the name with a terminating null byte (**dp->d_namlen+1**), rounded up to a 4-byte boundary.

```
#undef DIRSIZ
#define DIRSIZ(dp)    ((sizeof (struct direct) - (MAXNAMLEN+1)) + (((dp)->d_namlen+1 + 3) &~ 3))
struct   direct {
        u_long  d_ino;
        short   d_reclen;
        short   d_namlen;
        char    d_name[MAXNAMLEN + 1];
        /* typically shorter */
};
```

By convention, the first two entries in each directory are for '.' and '..'. The first is an entry for the directory itself. The second is for the parent directory. The meaning of '..' is modified for the root directory of the master file system ("/"), for which '..' has the same meaning as '.'.

SEE ALSO

getdirentries(2), **directory**(3V), **fs**(5)

NAME

dump, dumpdates – incremental dump format

SYNOPSIS

```
#include <sys/types.h>
#include <sys/inode.h>
#include <protocols/dumprestore.h>
```

DESCRIPTION

Tapes used by **dump** and **restore**(8) contain:

a header record
two groups of bit map records
a group of records describing directories
a group of records describing files

The format of the header record and of the first record of each description as given in the include file **<protocols/dumprestore.h>** is:

```
#define TP_BSIZE            1024
#define NTREC               10
#define HIGHDENSITYTREC     32
#define CARTRIDGETREC       63
#define TP_NINDIR           (TP_BSIZE/2)

#define TS_TAPE             1
#define TS_INODE            2
#define TS_BITS             3
#define TS_ADDR             4
#define TS_END              5
#define TS_CLRI             6
#define OFS_MAGIC           (int)60011
#define NFS_MAGIC           (int)60012
#define CHECKSUM            (int)84446
union u_spcl {
        char dummy[TP_BSIZE];
        struct                  s_spcl {
                                intc_type;
                                time_tc_date;
                                time_tc_ddate;
                                intc_volume;
                                daddr_tc_tapea;
                                ino_tc_inumber;
                                intc_magic;
                                intc_checksum;
                                structdinodec_dinode;
                                intc_count;
                                charc_addr[TP_NINDIR];
        } s_spcl;
} u_spcl;

#define spcl u_spcl.s_spcl

#define DUMPOUTFMT          "%-16s %c %s"/* for printf */
                            /* name, incno, ctime(date) */
#define DUMPINFMT           "%16s %c %[^\n]\n"/* inverse for scanf */
```

TP_BSIZE Size of file blocks on the dump tapes. Note: **TP_BSIZE** must be a multiple of **DEV_BSIZE.**

NTREC Default number of **TP_BSIZE** byte records in a physical tape block, changeable by the **b** option to **dump.**

HIGHDENSITYNTREC
Default number of **TP_BSIZE** byte records in a physical tape block on 6250 BPI or higher density tapes.

CARTRIDGETREC
Default number of **TP_BSIZE** records in a physical tape block on cartridge tapes.

TP_NINDIR Number of indirect pointers in a **TS_INODE** or **TS_ADDR** record. It must be a power of two.

The **TS_** entries are used in the **c_type** field to indicate what sort of header this is. The types and their meanings are as follows:

TS_TAPE Tape volume label

TS_INODE A file or directory follows. The **c_dinode** field is a copy of the disk inode and contains bits telling what sort of file this is.

TS_BITS A bit map follows. This bit map has a one bit for each inode that was dumped.

TS_ADDR A subrecord of a file description. See **c_addr** below.

TS_END End of tape record.

TS_CLRI A bit map follows. This bit map contains a zero bit for all inodes that were empty on the file system when dumped.

NFS_MAGIC All header records have this number in **c_magic.**

CHECKSUM Header records checksum to this value.

The fields of the header structure are as follows:

c_type The type of the header.

c_date The date the dump was taken.

c_ddate The date the file system was dumped from.

c_volume The current volume number of the dump.

c_tapea The current number of this (1024-byte) record.

c_inumber The number of the inode being dumped if this is of type **TS_INODE.**

c_magic This contains the value **MAGIC** above, truncated as needed.

c_checksum This contains whatever value is needed to make the record sum to **CHECKSUM.**

c_dinode This is a copy of the inode as it appears on the file system; see **fs**(5).

c_count The count of characters in **c_addr.**

c_addr An array of characters describing the blocks of the dumped file. A character is zero if the block associated with that character was not present on the file system, otherwise the character is non-zero. If the block was not present on the file system, no block was dumped; the block will be restored as a hole in the file. If there is not sufficient space in this record to describe all of the blocks in a file, **TS_ADDR** records will be scattered through the file, each one picking up where the last left off.

Each volume except the last ends with a tapemark (read as an end of file). The last volume ends with a **TS_END** record and then the tapemark.

The dump history is kept in the file **/etc/dumpdates**. It is an ASCII file with three fields separated by white space:

The name of the device on which the dumped file system resides.

The level number of the dump tape; see **dump**(8).

The date of the incremental dump in the format generated by **ctime**(3V).

DUMPOUTFMT is the format to use when using **printf**(3S) to write an entry to **/etc/dumpdates**; **DUMPINFMT** is the format to use when using **scanf**(3S) to read an entry from **/etc/dumpdates**.

FILES

/etc/dumpdates

SEE ALSO

fs(5), **types**(5), **dump**(8), **restore**(8)

NAME

environ – user environment

SYNOPSIS

extern char **environ;

DESCRIPTION

An array of strings called the 'environment' is made available by **execve**(2V) when a process begins. By convention these strings have the form '*name=value*'. The following names are used by various commands:

PATH The sequence of directory prefixes that **sh**(1), **time**(1V), **nice**(1), etc., apply in searching for a file known by an incomplete path name. The prefixes are separated by '**:**'. The **login**(1) process sets **PATH=:/usr/ucb:/bin:/usr/bin**.

HOME The name of the user's login directory, set by **login**(1) from the password file **/etc/passwd** (see **passwd**(5)).

TERM The type of terminal on which the user is logged in. This information is used by commands, such as **nroff**(1) or **plot**(1G), which may exploit special terminal capabilities. See **/etc/termcap** (**termcap**(5)) for a list of terminal types.

SHELL The path name of the user's login shell.

TERMCAP The string describing the terminal in TERM, or the name of the **termcap** file, see **termcap**(3X), **termcap**(5).

EXINIT A startup list of commands read by **ex**(1), **edit**, and **vi**(1).

USER
LOGNAME The user's login name.

TZ The name of the time zone that the user is located in. If **TZ** is not present in the environment, the system's default time zone, normally the time zone that the computer is located in, is used.

Further names may be placed in the environment by the *export* command and '*name=value*' arguments in **sh**(1), or by the **setenv** command if you use **csh**(1). Arguments may also be placed in the environment at the point of an **execve**(2V). It is unwise to conflict with certain **sh**(1) variables that are frequently exported by **.profile** files: **MAIL, PS1, PS2, IFS**.

SYSTEM V DESCRIPTION

The description of the variable **TERMCAP** does not apply to programs built in the System V environment.

FILES

/etc/passwd
etc/termcap

SEE ALSO

csh(1), **ex**(1), **login**(1), **nice**(1), **nroff**(1), **plot**(1G), **sh**(1), **time**(1V), **vi**(1), **execve**(2V), **getenv**(3V), **system**(3), **termcap**(3X), **passwd**(5), **termcap**(5)

NAME

ethers – Ethernet address to hostname database or NIS domain

DESCRIPTION

The **ethers** file contains information regarding the known (48 bit) Ethernet addresses of hosts on the Internet. For each host on an Ethernet, a single line should be present with the following information:

Ethernet-address official-host-name

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment extending to the end of line.

The standard form for Ethernet addresses is "*x:x:x:x:x:x*" where *x* is a hexadecimal number between 0 and ff, representing one byte. The address bytes are always in network order. Host names may contain any printable character other than a SPACE, TAB, NEWLINE, or comment character. It is intended that host names in the **ethers** file correspond to the host names in the **hosts**(5) file.

The **ether_line**( ) routine from the Ethernet address manipulation library, **ethers**(3N) may be used to scan lines of the **ethers** file.

FILES

/etc/ethers

SEE ALSO

ethers(3N), **hosts**(5)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

exports, xtab – directories to export to NFS clients

SYNOPSIS

/etc/exports

/etc/xtab

DESCRIPTION

The **/etc/exports** file contains entries for directories that can be exported to NFS clients. This file is read automatically by the **exportfs**(8) command. If you change this file, you must run **exportfs**(8) for the changes to affect the daemon's operation.

Only when this file is present at boot time does the **rc.local** script execute **exportfs**(8) and start the NFS file-system daemon, **nfsd**(8).

The **/etc/xtab** file contains entries for directories that are *currently* exported. This file should only be accessed by programs using **getexportent** (see **exportent**(3)). (Use the **–u** option of **exportfs** to remove entries from this file).

An entry for a directory consists of a line of the following form:

directory *–option*[,*option*]...

directory is the pathname of a directory (or file).

option is one of

ro Export the directory read-only. If not specified, the directory is exported read-write.

rw=*hostnames*[**:***hostname*]...
Export the directory read-mostly. Read-mostly means read-only to most machines, but read-write to those specified. If not specified, the directory is exported read-write to all.

anon=*uid*
If a request comes from an unknown user, use *uid* as the effective user ID. Note: root users (uid 0) are always considered "unknown" by the NFS server, unless they are included in the "root" option below. The default value for this option is –2. Setting "anon" to –1 disables anonymous access. Note: by default secure NFS will accept insecure requests as anonymous, and those wishing for extra security can disable this feature by setting "anon" to –1.

root=*hostnames*[**:***hostname*]...
Give root access only to the root users from a specified *hostname*. The default is for no hosts to be granted root access.

access=*client*[**:***client*]...
Give mount access to each *client* listed. A *client* can either be a hostname, or a netgroup (see **netgroup**(5)). Each *client* in the list is first checked for in the netgroup database, and then the hosts database. The default value allows any machine to mount the given directory.

secure Require clients to use a more secure protocol when accessing the directory.

A '#' (pound-sign) anywhere in the file indicates a comment that extends to the end of the line.

EXAMPLE

```
/usr        –access=clients          # export to my clients
/usr/local  # export to the world
/usr2       –access=hermes:zip:tutorial  # export to only these machines
/usr/sun    –root=hermes:zip         # give root access only to these
/usr/new    –anon=0                  # give all machines root access
```

```
/usr/bin        -ro                          # export read-only to everyone
/usr/stuff      -access=zip,anon=-3,ro       # several options on one line
```

FILES

/etc/exports
/etc/xtab
/etc/hosts
/etc/netgroup
rc.local

SEE ALSO

exportent(3), **hosts**(5), **netgroup**(5), **exportfs**(8), **nfsd**(8)

WARNINGS

You cannot export either a parent directory or a subdirectory of an exported directory that is *within the same filesystem*. It would be illegal, for instance, to export both **/usr** and **/usr/local** if both directories resided on the same disk partition.

NAME

ext_ports – external ports file for network printers, terminals, and modems

SYNOPSIS

/etc/ext_ports

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **ext_ports** external ports file is an ASCII file in the **/etc** directory on the Network Information Service (NIS) master server. **ext_ports** is used only by SNAP, and contains basic information about each printer, terminal, and modem on the network. This file contains a one-line entry for each device, and each field *must* be separated by a TAB character:

system:port type status baud model name #comment

system names the system to which the device is attached. This field contains only lower case and numeric characters, must start with a lower case character, and must not be longer than 32 characters.

port names the port in **/dev** on the *system*: **ttya** for the Sun386i serial port, **pp0** for the parallel port, and **ttym0** and **ttym1** for ports on an AT bus serial card.

type **printer**, **terminal**, or **modem**.

status indicates the device status. For terminals and printers, this can be **on** or **off**. An **off** status means the device is disabled from access by the SunOS operating system, but can still be accessed by DOS. For modems, this can be **in** to enable dialin, **out** to enable dialout, **in_out** to enable dialin and dialout, or **off**. An **off** status means the device is disabled from access by the SunOS operating system, but it can still be accessed by DOS.

baud is the baud rate.

model indicates the manufacturer or kind of device. For printers, this can be **epson**, **hp**, or **text**, for Epson and compatibles, HP Laserjet and compatibles, or for text-only printers. For terminals, this can be **vt100** or **wyse-50** for DEC VT-100 and compatibles or for Wyse WY-50 and compatibles. For modems, this can be **hayes** for Hayes and compatibles.

name is only used for unique naming of printers on the network. Up to 16 characters can be entered. This field is blank for terminals and modems — simply insert a TAB character.

#comment

can contain anything you want, up to a maximum of 96 characters.

EXAMPLE

In this example of an **ext_ports** file, the system vulcan has an epson printer attached to its parallel port, and a Wyse-50 terminal attached to its serial port, but with logins currently disabled. The system android has a VT100 attached to its serial port, with logins enabled. The system polaris has a **hayes** modem set for dialing out on an installed AT bus serial card.

```
vulcan:pp0      printer   on      9600   epson     lp   #Engineering lab
android:ttya    terminal  on      9600   vt100          #Reception
vulcan:ttya     terminal  off     9600   wyse-50        #Engineering lab
polaris:ttym0   modem     in_out  2400   hayes          #QA lab
```

FILES

/etc/ext_ports

SEE ALSO

snap(1), **vipw**(8)

Sun386i System and Network Administration,
Sun386i Advanced Administration

BUGS

The **/etc/ext_ports** file must be locked against simultaneous changes when it is edited; **vipw**(8) does the necessary locking.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

fbtab – framebuffer table

SYNOPSIS

/etc/fbtab

DESCRIPTION

The **/etc/fbtab** file contains information that is used by **login**(1), **getty**(8) and the window system (for example, **sunview**(1)) to change the owner, group, and permissions of window system devices upon logging into or out of a console device. By default, *all* lines in this file are commented out. That is, all window security is disabled. To enable window security, edit **/etc/fbtab**, log out, and log back in. You *must* edit this file before window security can be enabled.

The owner and group of the devices listed in **/etc/fbtab** are set to the owner and group of the console. The permissions are set as specified in **/etc/fbtab**. As in the example below, **0600** is the recommended permissions for normal security.

Fields are separated by TAB and/or SPACE characters. Blank lines and comments can appear anywhere in the file; comments are delimited by '**#**' and a NEWLINE.

The first field specifies the name of a console device (for example, **/dev/console**). The second field specifies the permissions to which the devices in the *device_list* field (third field) will be set. A *device_list* is a colon-separated list of device names (the full pathname is required).

Once the devices are owned by the user, their permissions and ownership can be changed using **chmod**(1V) and **chown**(8), as with any other user-owned file.

EXAMPLES

The following example entry in the **/etc/fbtab** file enables normal window security:

```
/dev/console 0600    /dev/kbd:/dev/mouse
/dev/console 0600    /dev/fb:/dev/bwone0:/dev/bwtwo0
/dev/console 0600    /dev/cgone0:/dev/cgtwo0:/dev/cgthree0:/dev/cgfour0
/dev/console 0600    /dev/cgsix0:/dev/cgeight0:/dev/cgnine0
/dev/console 0600    /dev/gpone0a:/dev/gpone0b:/dev/gpone0c:/dev/gpone0d
```

This entry specifies that upon login to **/dev/console**, the owner, group and permissions of all supported devices will be set to the user's username, the user's group and **0600**, respectively. You need only specify the devices supported by your configuration. Upon logout, the owner and group of these devices will be reset to **root** and **wheel**. The permissions remain as set in the **/etc/fbtab** file.

SEE ALSO

login(1), **sunview**(1), **sv_acquire**(1), **getty**(8)

NAME

fcntl – file control options

SYNOPSIS

#include <fcntl.h>

DESCRIPTION

The **fcntl**(2V) function provides for control over open files. This include file describes *requests* and *arguments* to **fcntl** and **open**(2V) as shown below:

```
/*              @ (#)fcntl.h 1.2 83/12/08 SMI; from UCB 4.2 83/09/25*/
/*
* Flag values accessible to open(2V) and fcntl(2)
*  (The first three can only be set by open)
*/
#define         O_RDONLY                0
#define         O_WRONLY                1
#define         O_RDWR                  2
#define         O_NDELAY                FNDELAY     /* Non-blocking I/O */
#define         O_APPEND                FAPPEND     /* append (writes guaranteed at the end) */
#ifndef         F_DUPFD
/* fcntl(2) requests */
#define         F_DUPFD                 0           /* Duplicate fildes */
#define         F_GETFD                 1           /* Get fildes flags */
#define         F_SETFD                 2           /* Set fildes flags */
#define         F_GETFL                 3           /* Get file flags */
#define         F_SETFL                 4           /* Set file flags */
#define         F_GETOWN                5           /* Get owner */
#define         F_SETOWN                6           /* Set owner */
/* flags for F_GETFL, F_SETFL— copied from <sys/file.h> */
#define         FNDELAY                             00004/* non-blocking reads */
#define         FAPPEND                             00010/* append on each write */
#define         FASYNC                              00100/* signal pgrp when data ready */
#endif
```

SEE ALSO

fcntl(2V), **open**(2V)

NAME

fs, inode – format of a 4.2 (ufs) file system volume

SYNOPSIS

#include <sys/types.h>
#include <ufs/fs.h>
#include <ufs/inode.h>

DESCRIPTION

Standard 4.2 (ufs) file system storage volumes have a common format for certain vital information. Every such volume is divided into a certain number of blocks. The block size is a parameter of the file system. Sectors 0 to 15 contain primary and secondary bootstrapping programs.

The actual file system begins at sector 16 with the *super-block*. The layout of the super block is defined by the include file **<ufs/fs.h>**

Each disk drive contains some number of file systems. A file system consists of a number of cylinder groups. Each cylinder group contains inodes and data.

A file system is described by its super-block, which in turn describes the cylinder groups. The super-block is critical data and is replicated in each cylinder group to protect against catastrophic loss. This is done at file system creation time and the critical super-block data does not change, so the copies need not be referenced further unless disaster strikes.

Addresses stored in inodes are capable of addressing fragments of "blocks." File system blocks of at most size **MAXBSIZE** can be optionally broken into 2, 4, or 8 pieces, each of which is addressable; these pieces may be **DEV_BSIZE**, or some multiple of a **DEV_BSIZE** unit.

Large files consist of exclusively large data blocks. To avoid undue wasted disk space, the last data block of a small file is allocated as only as many fragments of a large block as are necessary. The file system format retains only a single pointer to such a fragment, which is a piece of a single large block that has been divided. The size of such a fragment is determinable from information in the inode, using the '**blksize(fs, ip, lbn)**' macro.

The file system records space availability at the fragment level; to determine block availability, aligned fragments are examined.

The root inode is the root of the file system. Inode 0 cannot be used for normal purposes and historically bad blocks were linked to inode 1, thus the root inode is 2 (inode 1 is no longer used for this purpose, however numerous dump tapes make this assumption, so we are stuck with it). The *lost+found* directory is given the next available inode when it is initially created by **mkfs**(8).

fs_minfree gives the minimum acceptable percentage of file system blocks which may be free. If the freelist drops below this level only the super-user may continue to allocate blocks. This may be set to 0 if no reserve of free blocks is deemed necessary, however severe performance degradations will be observed if the file system is run at greater than 90% full; thus the default value of **fs_minfree** is 10%.

Empirically the best trade-off between block fragmentation and overall disk utilization at a loading of 90% comes with a fragmentation of 4, thus the default fragment size is a fourth of the block size.

Cylinder group related limits: Each cylinder keeps track of the availability of blocks at different rotational positions, so that sequential blocks can be laid out with minimum rotational latency. **fs_nrpos** is the number of rotational positions which are distinguished. With the default **fs_nrpos** of 8 the resolution of the summary information is 2ms for a typical 3600 rpm drive.

fs_rotdelay gives the minimum number of milliseconds to initiate another disk transfer on the same cylinder. It is used in determining the rotationally optimal layout for disk blocks within a file; the default value for **fs_rotdelay** is 2ms.

Each file system has a statically allocated number of inodes. An inode is allocated for each **NBPI** bytes of disk space. The inode allocation strategy is extremely conservative.

MINBSIZE is the smallest allowable block size. With a **MINBSIZE** of 4096 it is possible to create files of size 2^32 with only two levels of indirection. **MINBSIZE** must be big enough to hold a cylinder group block, thus changes to **(struct cg)** must keep its size within **MINBSIZE.** Note: super blocks are never more than size **SBSIZE.**

The path name on which the file system is mounted is maintained in **fs_fsmnt.** **MAXMNTLEN** defines the amount of space allocated in the super block for this name. The limit on the amount of summary information per file system is defined by **MAXCSBUFS.** It is currently parameterized for a maximum of two million cylinders.

Per cylinder group information is summarized in blocks allocated from the first cylinder group's data blocks. These blocks are read in from **fs_csaddr** (size **fs_cssize**) in addition to the super block.

Note: sizeof **(struct csum)** must be a power of two in order for the **fs_cs** macro to work.

inode: The inode is the focus of all file activity in the file system. There is a unique inode allocated for each active file, each current directory, each mounted-on file, text file, and the root. An inode is "named" by its device/i-number pair. For further information, see the include file **<ufs/inode.h>**.

SEE ALSO

mkfs(8)

NAME

fspec – format specification in text files

DESCRIPTION

It is sometimes convenient to maintain text files on the operating system with non-standard tab stop settings, (that is, tab stops that are not set at every eighth column). Such files must generally be converted to a standard format, frequently by replacing all TAB characters with the appropriate number of SPACE characters, before they can be processed by operating system commands. A format specification occurring in the first line of a text file specifies how TAB characters are to be expanded in the remainder of the file.

A format specification consists of a sequence of parameters separated by blanks and surrounded by the brackets **<:** and **:>**. Each parameter consists of a keyletter, possibly followed immediately by a value. The following parameters are recognized:

t *tabs* The **t** parameter specifies the tab stop settings for the file. The value of *tabs* must be one of the following:

- A list of column numbers separated by commas, indicating tab stops set at the specified columns;
- A '–' followed immediately by an integer *n*, indicating tab stops set at intervals of *n* columns, that is, at 1+*n*, 1+2*n*, and so on;
- A '–' followed by the name of a "canned" tab stop specification.

Up to 40 numbers are allowed in a comma-separated list of tab stop settings. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Thus, the formats **t1, 10, 20, 30** and **t1, 10, +10, +10** are considered identical.

Standard tab stops are specified by **t–8**, or equivalently, **t1, 9, 17, 25,** etc. This is the tab stop setting that most operating system utilities assume, and is the most likely setting to be found at a terminal. The specification **t–0** specifies no tab stops at all.

The "canned" tab stops specifications that are recognized are as follows:

a 1, 10, 16, 36, 72
Assembler, IBM S/370, first format

a2 1, 10, 16, 40, 72
Assembler, IBM S/370, second format

c 1, 8, 12, 16, 20, 55
COBOL, normal format

c2 1, 6, 10, 14, 49
COBOL compact format (columns 1-6 omitted). Using this code, the first typed character corresponds to card column 7, one space gets you to column 8, and a TAB reaches column 12. Files using this tab stop setup should include a format specification as follows:

<:t–c2 m6 s66 d:>

c3 1, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62, 67
COBOL compact format (columns 1-6 omitted), with more tab stops than **c2**. This is the recommended format for COBOL. The appropriate format specification is:

<:t–c3 m6 s66 d:>

f 1, 7, 11, 15, 19, 23
FORTRAN

p 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
PL/I

s 1, 10, 55
SNOBOL

u 1, 12, 20, 44
UNIVAC 1100 Assembler

s *size* The **s** parameter specifies a maximum line size. The value of **size** must be an integer. Size checking is performed after TAB characters have been expanded, but before the margin is prepended.

m *margin*
The **m** parameter specifies a number of SPACE characters to be prepended to each line. The value of *margin* must be an integer.

d The **d** parameter takes no value. Its presence indicates that the line containing the format specification is to be deleted from the converted file.

e The **e** parameter takes no value. Its presence indicates that the current format is to prevail only until another format specification is encountered in the file.

Default values, which are assumed for parameters not supplied, are **t–8** and **m0**. If the **s** parameter is not specified, no size checking is performed. If the first line of a file does not contain a format specification, the above defaults are assumed for the entire file. The following is an example of a line containing a format specification:

*** <:t5,10,15 s72:> ***

If a format specification can be disguised as a comment, it is not necessary to code the **d** parameter.

SEE ALSO

ed(1), **tabs**(1V)

NAME

fstab, mtab – static filesystem mounting table, mounted filesystems table

SYNOPSIS

/etc/fstab

/etc/mtab

DESCRIPTION

The **/etc/fstab** file contains entries for filesystems and disk partitions to mount using the **mount**(8) command, which is normally invoked by the **rc.boot** script at boot time. This file is used by various utilities that mount, unmount, check the consistency of, dump, and restore file systems. It is also used by the system itself when locating the swap partition.

The **/etc/mtab** file contains entries for filesystems *currently* mounted, and is read by programs using the routines described in **getmntent**(3). **umount** (see **mount**(8)) removes entries from this file.

Each entry consists of a line of the form:

filesystem directory type options freq pass

filesystem is the pathname of a block-special device, the name of a remote filesystem in *host*:*pathname* form, or the name of a "swap file" made with **mkfile**(8).

directory is the pathname of the directory on which to mount the filesystem.

type is the filesystem type, which can be one of:

- **4.2** to mount a block-special device
- **lo** to loopback-mount a file system
- **nfs** to mount an exported NFS filesystem
- **swap** to indicate a swap partition
- **ignore** to have the **mount** command ignore the current entry (good for noting disk partitions that are not being used)
- **rfs** to mount an RFS filesystem
- **tmp** filesystem in virtual memory
- **hsfs** to mount an ISO 9660 Standard or High Sierra Standard CD-ROM filesystem

options contains a comma-separated list (no spaces) of mounting options, some of which can be applied to all types of filesystems, and others which only apply to specific types.

4.2 options:

quota | noquota Disk quotas are enforced or not enforced. The default is **noquota**.

nfs options:

- **bg | fg** If the first attempt fails, retry in the background, or, in the foreground.
- **noquota** Prevent **quota**(1) from checking whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.
- **retry=***n* The number of times to retry the mount operation.
- **rsize=***n* Set the read buffer size to *n* bytes.
- **wsize=***n* Set the write buffer size to *n* bytes.
- **timeo=***n* Set the NFS timeout to *n* tenths of a second.
- **retrans=***n* The number of NFS retransmissions.
- **port=***n* The server IP port number.
- **soft | hard** Return an error if the server does not respond, or continue the retry request until the server responds.
- **intr** Allow keyboard interrupts on hard mounts.
- **secure** Use a more secure protocol for NFS transactions.

acregmin=*n*
Hold cached attributes for at least *n* seconds after file modification.
acregmax=*n*
Hold cached attributes for no more than *n* seconds after file modification.
acdirmin=*n*
Hold cached attributes for at least *n* seconds after directory update.
acdirmax=*n*
Hold cached attributes for no more than *n* seconds after directory update.
actimeo=*n*
Set *min* and *max* times for regular files and directories to *n* seconds.
noac Suppress attribute caching.

Regular defaults are:
fg,retry=10000,timeo=7,retrans=3,port=NFS_PORT,hard,
acregmin=3,acregmax=60,acdirmin=30,acdirmax=60

actimeo has no default; it sets **acregmin**, **acregmax**, **acdirmin** and **acdirmax**

Defaults for **rsize** and **wsize** are set internally by the system kernel.

rfs options:

bg | fg If the first attempt fails, retry in the background, or, in the foreground.
retry=*n* The number of times to retry the mount operation.

Defaults are the same as for NFS.

Common options:

ro | rw mount either read-only or read-write
suid | nosuid
setuid execution allowed or disallowed
grpid Create files with BSD semantics for propagation of the group ID. With this option, files inherit the group ID of the directory in which they are created, regardless of the directory's setgid bit.
noauto Do not mount this file system automatically (using '**mount –a**').

freq is the interval (in days) between dumps.

pass is the **fsck**(8) pass in which to check the partition. Filesystems with *pass* 0 are not checked. Filesystems with the pass 1 are checked sequentially. In general, the root filesystem should be checked in pass 1, with others checked in higher (later) passes. For passes higher than 1, multiple filesystems in the same pass are checked simultaneously.

A hash-sign (#) as the first character indicates a comment line which is ignored by routines that read this file. The order of records in **/etc/fstab** is important because **fsck**, **mount**, and **umount** process the file sequentially; an entry for a file system must appear *after* the entry for any file system it is to be mounted on top of.

EXAMPLES

In this example, two partitions on the local disk are **4.2** mounted. Several **/export** directories are loopback mounted to appear in the traditional file system locations on the local system. The **/home/user** directory is hard mounted read-write over the NFS, along with additional swap space in the form of a mounted swap file (see *System and Network Administration* for details on adding swap space):

```
/dev/xy0a / 4.2 rw,noquota 1 1
/dev/xy0b /usr 4.2 rw,noquota 1 1
/export/tmp/localhost /tmp lo rw 0 0
/export/var/localhost /var lo rw 0 0
/export/cluster/sun386.sunos4.0.1 /usr/cluster lo rw 0 0
/export/local/sun386 /usr/local lo rw 0 0
```

```
example:/home/user /home/user nfs rw,hard,fg 0 0
/export/swap/myswap swap swap rw 0 0
```

FILES

/etc/fstab
/etc/mtab

SEE ALSO

swapon(2), **getmntent**(3), **lofs**(4S), **fsck**(8), **mkfile**(8), **mount**(8), **quotacheck**(8), **quotaon**(8), **swapon**(8)

System and Network Administration

NAME

ftpusers – list of users prohibited by FTP

SYNOPSIS

/etc/ftpusers

DESCRIPTION

ftpusers contains a list of users who cannot access this system using the File Transfer Protocol (FTP). **ftpusers** contains one user name per line.

If this file is missing, the list of users is considered to be empty, so that any user may use FTP to access the system if the other criteria for access are met (see **ftpd**(8C)).

SEE ALSO

ftp(1C), **ftpd**(8C)

NAME

gettytab – terminal configuration data base

SYNOPSIS

/etc/gettytab

DESCRIPTION

gettytab is a simplified version of the **termcap**(5) data base used to describe terminal lines. The initial terminal login process **getty**(8) accesses the **gettytab** file each time it starts, allowing simpler reconfiguration of terminal characteristics. Each entry in the data base is used to describe one class of terminals.

There is a default terminal class, **default**, that is used to set global defaults for all other classes. That is, the **default** entry is read, then the entry for the class required is used to override particular settings.

CAPABILITIES

Refer to **termcap**(5) for a description of the file layout. The *Default* column below lists defaults obtained if there is no entry in the table obtained, nor one in the special default table.

Name	*Type*	*Default*	*Description*
ab	bool	false	read a \r first and guess the baud rate from it
ap	bool	false	terminal uses 7 bits, any parity
bd	num	0	backspace delay
bk	str	0377	alternate end of line character (input break)
cb	bool	false	use crt backspace mode
cd	num	0	carriage-return delay
ce	bool	false	use crt erase algorithm
ck	bool	false	use crt kill algorithm
cl	str	NULL	screen clear sequence
co	bool	false	console - add NEWLINE after login prompt
de	num	0	delay before first prompt is printed (seconds)
ds	str	^Y	delayed suspend character
dx	bool	false	set DECCTLQ
ec	bool	false	leave echo OFF
ep	bool	false	terminal uses 7 bits, even parity
er	str	^?	erase character
et	str	^D	end of text (EOF) character
ev	str	NULL	initial environment
f0	num	unused	tty mode flags to write messages
f1	num	unused	tty mode flags to read login name
f2	num	unused	tty mode flags to leave terminal as
fd	num	0	form-feed (vertical motion) delay
fl	str	^O	output flush character
hc	bool	false	do NOT hangup line on last close
he	str	NULL	hostname editing string
hn	str	hostname	hostname
ht	bool	false	terminal has real tabs
ig	bool	false	ignore garbage characters in login name
im	str	NULL	initial (banner) message
in	str	^C	interrupt character
is	num	unused	input speed
kl	str	^U	kill character
lc	bool	false	terminal has lower case
lm	str	login:	login prompt
ln	str	^V	"literal next" character
lo	str	/usr/bin/login	program to exec when name obtained

ms	str	NULL	list of terminal modes to set or clear
m0	str	NULL	set modes that apply at the same time as those set by f0
m1	str	NULL	set modes that apply at the same time as those set by f1
m2	str	NULL	set modes that apply at the same time as those set by f2
nd	num	0	NEWLINE (LINEFEED) delay
nl	bool	false	terminal has (or might have) a NEWLINE character
nx	str	default	next table (for auto speed selection)
op	bool	false	terminal uses 7 bits, odd parity
os	num	unused	output speed
p8	bool	false	terminal uses 8 bits, no parity
pc	str		pad character
pe	bool	false	use printer (hard copy) erase algorithm
pf	num	0	delay between first prompt and following flush (seconds)
ps	bool	false	line connected to a MICOM port selector
qu	str	^	quit character
rp	str	^R	line retype character
rw	bool	false	do NOT use RAW for input, use CBREAK
sp	num	0	line speed (input and output)
su	str	^Z	suspend character
tc	str	none	table continuation
td	num	0	tab delay
to	num	0	timeout (seconds)
tt	str	NULL	terminal type (for environment)
ub	bool	false	do unbuffered output (of prompts etc)
uc	bool	false	terminal is known upper case only
we	str	^W	word erase character
xc	bool	false	do NOT echo control chars as ^X
xf	str	^S	XOFF (stop output) character
xn	str	^Q	XON (start output) character

If no line speed is specified, speed will not be altered from that which prevails when **getty** is entered. Specifying an input or output speed overrides line speed for stated direction only. If **ab** is specified, **getty** will initially read a character from the tty, assumed to be a carriage return, and will attempt to figure out the baud rate based on what the character appears as. It will then look for a table entry for that baud rate; if the line appears to be a 300 baud line, it will look for an entry **300-baud**, if it appears to be a 1200 baud line, it will look for an entry **1200-baud**, etc..

Terminal modes to be used for the output of the message, for input of the login name, and to leave the terminal set as upon completion, are derived from the Boolean flags specified. If the derivation should prove inadequate, any (or all) of these three may be overridden with one of the **f0**, **f1**, or **f2** numeric specifications, which can be used to specify (usually in octal, with a leading '0') the exact values of the flags. Local (new tty) flags are set in the top 16 bits of this (32 bit) value.

The **ms** field can be used to specify modes to be set and cleared. These modes are specified as **stty**(1V) modes; any mode supported by **stty** may be specified, except for the baud rate which must be specified with the **br** field. This permits modes not supported by the older terminal interface described in **ttcompat**(4M) to be set or cleared. Thus, to set the terminal port to which the printer is attached to even parity, TAB expansion, no NEWLINE to RETURN/LINEFEED translation, and RTS/CTS flow control enabled, do:

:ms=evenp,-tabs,nl,crtscts:

The **m0**, **m1**, and **m2** fields can be used to set modes which only apply concurrently with those set by **f0**, **f1**, and **f2**, respectively. The modes specified by **ms**, **m0**, **m1**, and **m2** are applied *after* the modes specified by other existing capabilities.

Should **getty** receive a null character (presumed to indicate a line break) it will restart using the table indicated by the **nx** entry. If there is none, it will re-use its original table.

Delays are specified in milliseconds, the nearest possible delay available in the tty driver will be used. Should greater certainty be desired, delays with values 0, 1, 2, and 3 are interpreted as choosing that particular delay algorithm from the driver.

The **cl** screen clear string may be preceded by a (decimal) number of milliseconds of delay required (as with **termcap**(5)). This delay is simulated by repeated use of the pad character **pc**.

The initial message, and login message, **im** and **lm** may include the character sequence **%h** or **%t** to obtain the hostname or tty name respectively. (**%%** obtains a single '**%**' character.) The hostname is normally obtained from the system, but may be set by the **hn** table entry. In either case it may be edited with **he**. The **he** string is a sequence of characters, each character that is neither '**@**' nor '**#**' is copied into the final hostname. A '**@**' in the **he** string, copies one character from the real hostname to the final hostname. A '**#**' in the **he** string, skips the next character of the real hostname. Surplus '**@**' and '**#**' characters are ignored.

When **getty** execs the login process, given in the **lo** string (usually **/usr/bin/login**), it will have set the environment to include the terminal type, as indicated by the **tt** string (if it exists). The **ev** string, can be used to enter additional data into the environment. It is a list of comma separated strings, each of which will presumably be of the form *name=value*.

If a non-zero timeout is specified, with **to**, then **getty** will exit within the indicated number of seconds, either having received a login name and passed control to *login*, or having received an alarm signal, and exited. This may be useful to hangup dial in lines.

Output from **getty** is even parity unless **op** or **p8** is specified. **op** may be specified with **ap** to allow any parity on input, but generate odd parity output. Note: this only applies while **getty** is being run, terminal driver limitations prevent a more complete implementation. **getty** does not check parity of input characters in RAW mode.

FILES

/etc/gettytab

SEE ALSO

termcap(5), **getty**(8)

NAME

group – group file

SYNOPSIS

/etc/group

DESCRIPTION

The **group** file contains a one-line entry for each group recognized by the system, of the form:

groupname:*password*:*gid*:*user-list*

where:

groupname is the name of the group.

gid is the group's numerical ID within the system; it must be unique.

user-list is a comma-separated list of users allowed in the group.

If the password field is empty, no password is demanded. The **group** file is an ASCII file. Because of the encrypted passwords, the **group** file can and does have general read permission, and can be used as a mapping of numerical group IDs to group names.

A group entry beginning with a '+' (plus sign), means to incorporate an entry or entries from the Network Information Service (NIS) A '+' on a line by itself means to insert the entire contents of the NIS group file at that point in the file. An entry of the form: '+*groupname*' means to insert the entry (if any) for **groupname**. If a '+' entry has a non-empty *password* or *user-list* field, the contents of that field override the corresponding field from the NIS service. The *gid* field cannot be overridden in this way.

An entry of the form: –*groupname* indicates that the group is disallowed. All subsequent entries for the indicated *groupname*, whether originating from the NIS service, or the local **group** file, are ignored.

Malformed entries cause routines that read this file to halt, in which case group assignments specified further along are never made. To prevent this from happening, use **grpck**(8) to check the **/etc/group** database from time to time.

Sun386i systems uses the following group IDs as program privileges:

operator	5	Privilege to do backup as root.
accounts	11	Privilege to update user accounts.
networks	12	Privilege to change network configuration.
devices	13	Privilege to modify printer, terminal, or modem configurations.

On all Sun systems, SunOS uses group ID 0 as privilege to run **su**(1V).

EXAMPLE

Here is a sample group file when the **group.adjunct** file does not exist:

```
primary:q.mJzTnu8icF.:10:fred,mary
+myproject:::bill,steve
+:
```

Here is a sample group file when the **group.adjunct** file does exist:

```
primary:#$primary:10:fred,mary
+myproject:::bill,steve
+:
```

If these entries appear at the end of a group file, then the group *primary* will have members **fred** and **mary**, and a group ID of **10**. The group *myproject* will have members **bill** and **steve**, and the password and group ID of the NIS entry for the group **myproject**. All groups listed in the NIS service are pulled in and placed after the entry for **myproject**.

FILES

/etc/group

SEE ALSO

passwd(1), **su**(1V), **getgroups**(2V), **crypt**(3), **initgroups**(3), **group.adjunct**(5), **passwd**(5), **grpck**(8V)

NOTES

SunOS releases prior to SunOS 4.0, permitted a user to belong to no more then eight groups at a time. A user who belongs to more than eight groups may have trouble using the RPC service (and therefore NFS) to communicate with machines running older releases. In such cases, RPC complains of an "Authentication Error".

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

BUGS

The **passwd**(1) command will not change group passwords.

NAME

group.adjunct – group security data file

SYNOPSIS

/etc/security/group.adjunct

DESCRIPTION

The **group.adjunct** file contains the following information for each group:

groupname:*password*

groupname The group's name in the system; it must be unique.

password The encrypted password, formerly field two of the **/etc/group** file.

The **group.adjunct** file is in ASCII. Fields are separated by a colon, and each group is separated from the next by a NEWLINE.

A **group.adjunct** file can have a line beginning with a '+' (plus sign), which means to incorporate entries from the Network Information Service (NIS). There are two styles of '+' entries: all by itself, '+' means to insert the entire contents of the **group.adjunct** NIS file at that point; *+name* means to insert the entry (if any) for *name* from the NIS service at that point. If a '+' entry has a non-null password, the contents of that field will override what is contained in the NIS service.

FILES

/etc/group

SEE ALSO

crypt(3), **getgraent**(3), **getgrent**(3V), **group**(5)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

help – help file format

SYNOPSIS

/usr/lib/help/*

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

Each SunView application using the **help** feature has a simple ASCII file in **/usr/lib/help** with the name *application-name*.**info**.

This file contains the text of help messages for each SunView object within that program. Each help message is separated in the file by a line beginning with a colon and identified by a keyword which matches the HELP_DATA attribute of the SunView object.

The first character of each line in the file may be:

#	comment line
:	keyword line
any other	1-32 help text lines

If the line is a keyword line, it has the following structure—

:keyword[s]:datastring [pagenumber]NEWLINE

keyword is a 1-65 character keyword
--any displayable characters may be used
--several keywords may be present
--keywords are separated by 1-or-more blanks

datastring is 1-256 ASCII bytes, and describes the path of the data files for help_viewer, relative to /usr/lib/help.

pagenumber is an optional page number within the help_viewer data file.

The help text which follows the :keyword line will be displayed in an Alert Box when help is requested for one of the keywords by pressing the help key.

The datastring will be sent (by RPC) to the **help_viewer** procedure when the user selects the More Help box in the Alert Box window.

EXAMPLE

Here is part of a typical help file, called **mailtool.info**.

:abort
Abort button

o Quits the Mail application (click left on button). Tentative message deletions do not become permanent.

o Provides a menu of Abort options (click right on button).

:cancel:mailtool/Writing_and_Sending_Mail 1
Cancel button

o Closes the message composition window without sending message (click left on button).

o Provides a menu of Cancel options (click right on button).

Pressing the help key while in the cancel or abort buttons triggers the display of the corresponding text. The words *cancel* and *abort* in this file are the keywords. In the case of abort, there is no More Help available. For cancel, More Help is available and it is stored in the first page of the **Writing_and_Sending_Mail** file in the mailtool directory.

FILES

/usr/lib/help/* files for the pop-up help facility

SEE ALSO

help_viewer(1), **help_viewer**(5)

Sun386i Developer's Guide

NAME

help_viewer – help viewer file format

SYNOPSIS

/usr/lib/help/*/*

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **help_viewer** reads files of various types. The Top Level list of applications documented is **/usr/lib/help/Top_Level**. The Master Index shown at the top level is **/usr/lib/help/Master_Index**. These files are FrameMaker files. To add or remove a heading from this list, use FrameMaker (1.1 or later).

Each directory within **/usr/lib/help** that corresponds to a SunView application name contains detailed information about that application. These are also FrameMaker files. The ***.rf** files are rasterfiles, of standard image format created by FrameMaker. These are the pictures that are interleaved into the text.

The **Frame/** subdirectory of **/usr/lib/help** contains topic, contents, and index templates which can be used to create new Help Viewer handbooks. The **Interleaf/** subdirectory contains Interleaf templates, fonts, and initialization files.

FILES

/usr/lib/help/*/*

SEE ALSO

help(5), **help_viewer**(1)

NAME

hosts – host name data base

SYNOPSIS

/etc/hosts

DESCRIPTION

The **hosts** file contains information regarding the known hosts on the TCP/IP. For each host a single line should be present with the following information:

Internet-address official-host-name aliases

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official host data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown hosts.

Network addresses are specified in the conventional '.' notation using the **inet_addr** ( ) routine from the Internet address manipulation library, **inet**(3N). Host names may contain any printable character other than an upper case character, a field delimiter, NEWLINE, or comment character.

EXAMPLE

Here is a typical line from the **/etc/hosts** file:

192.9.1.20 gaia # John Smith

FILES

/etc/hosts

SEE ALSO

gethostent(3N), **inet**(3N)

NAME

hosts.equiv, .rhosts – trusted remote hosts and users

DESCRIPTION

The **/etc/hosts.equiv** and **.rhosts** files provide the "remote authentication" database for **rlogin**(1C), **rsh**(1C), **rcp**(1C), and **rcmd**(3N). The files specify remote hosts and users that are considered *trusted*. Trusted users are allowed to access the local system *without supplying a password.* The library routine **ruserok()** (see **rcmd**(3N)) performs the authentication procedure for programs by using the **/etc/hosts.equiv** and **.rhosts** files. The **/etc/hosts.equiv** file applies to the entire system, while individual users can maintain their own **.rhosts** files in their home directories.

These files *bypass* the standard password-based user authentication mechanism. To maintain system security, care must be taken in creating and maintaining these files.

The remote authentication procedure determines whether a particular remote user from a particular remote host should be allowed to access the local system as a (possibly different) particular local user. This procedure first checks the **/etc/hosts.equiv** file and then checks the **.rhosts** file in the home directory of the local user as whom access is being attempted. Entries in these files can be of two forms. *Positive* entries explicitly *allow* access, while *negative* entries explicitly *deny* access. The authentication succeeds as soon as a matching positive entry is found. The procedure fails when a matching negative entry is found, or if no matching entries are found in either file. The order of entries, therefore, can be important: If the files contain both matching positive and negative entries, the entry that appears first will prevail. The **rsh**(1C) and **rcp**(1C) programs fail if the remote authentication procedure fails. The **rlogin** program will fall back to the standard password-based login procedure if the remote authentication fails.

Both files are formatted as a list of one-line entries. Each entry has the form:

hostname [*username*]

Negative entries are differentiated from positive entries by a '-' character preceding either the *hostname* or *username* field.

Positive Entries

If the form:

hostname

is used, then users from the named host are trusted. That is, they may access the system with the same user name as they have on the remote system. This form may be used in both the **/etc/hosts.equiv** and **.rhosts** files.

If the line is in the form:

hostname username

then the named user from the named host can access the system. This form may be used in individual **.rhosts** files to allow remote users to access the system *as a different local user.* If this form is used in the **/etc/hosts.equiv** file, the named remote user will be allowed to access the system as *any* local user.

Netgroups(5) can be used in either the *hostname* or *username* fields to match a number of hosts or users in one entry. The form:

+@netgroup

allows access from all hosts in the named netgroup. When used in the *username* field, netgroups allow a group of remote users to access the system as a particular local user. The form:

hostname +@netgroup

allows all of the users in the named netgroup from the named host to access the system as the local user. The form:

+@netgroup1 +@netgroup2

allows the users in *netgroup2* from the hosts in *netgroup1* to access the system as the local user.

The special character '+' can be used in place of either *hostname* or *username* to match any host or user. For example, the entry

+

will allow a user from any remote host to access the system with the same username. The entry

+ *username*

will allow the named user from any remote host to access the system. The entry

hostname +

will allow any user from the named host to access the system as the local user.

Negative Entries

Negative entries are preceded by a '-' sign. The form:

-*hostname*

will disallow all access from the named host. The form:

−*@netgroup*

means that access is explicitly disallowed from all hosts in the named netgroup. The form:

hostname −username

disallows access by the named user only from the named host, while the form:

+ −*@netgroup*

will disallow access by all of the users in the named netgroup from all hosts.

FILES

/etc/hosts.equiv
~/.rhosts

NOTES

Hostnames in **/etc/hosts.equiv** and **.rhosts** files must be the "official" name of the host, not one of its nicknames.

Root access is handled as a special case. Only the */.rhosts* file is checked when the access is being attempted for root. To help maintain system security, the */etc/hosts.equiv* file is not checked.

As a security feature, the *.rhosts* file must be owned by the user as whom access is being attempted.

Positive entries in */etc/hosts.equiv* that include a *username* field (either an individual named user, a netgroup, or '+' sign) should be used only with extreme caution. Because */etc/hosts.equiv* applies systemwide, these entries allow one or a group of remote users to access the system *as any local user*. This can be the source of a security hole.

SEE ALSO

rlogin(1C), **rsh**(1C), **rcp**(1C), **rcmd**(3N), **hosts**(5), **netgroup**(5), **passwd**(5)

NAME

indent.pro – default options for indent

DESCRIPTION

The **.indent.pro** file in either the current or home directory contains default command line options for the **indent**(1) program. It is a text file that contains space-separated command line options. For a description of these options, see **indent**(1).

Explicit command line options override options taken from **.indent.pro**.

Here is a sample **.indent.pro** file:

–bap –nbad –nbbb –bc –br –cdb –nce
–fc1 –ip –lp –npcs –psl –sc –nsob –cli0
–di12 –l79 –i4 –d0 –c33

FILES

./.indent.pro
~/.indent.pro

SEE ALSO

indent(1)

NAME

inetd.conf – Internet servers database

DESCRIPTION

The **inetd.conf** file contains the list of servers that **inetd**(8C) invokes when it receives an Internet request over a socket. Each server entry is composed of a single line of the form:

service-name socket-type protocol wait-status uid server-program server-arguments

Fields can be separated by either spaces or TAB characters. A '#' (pound-sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search this file.

service-name is the name of a valid service listed in the file **/etc/services**. For RPC services, the value of the *service-name* field consists of the RPC service name, followed by a slash and either a version number or a range of version numbers (for example, **mountd/1**).

socket-type can be one of:

stream	for a stream socket,
dgram	for a datagram socket,
raw	for a raw socket,
rdm	for a "reliably delivered message" socket, or
seqpacket	for a sequenced packet socket.

protocol must be a recognized protocol listed in the file **/etc/protocols**. For RPC services, the field consists of the string "rpc" followed by a slash and the name of the protocol (for example, **rpc/udp** for an RPC service using the UDP protocol as a transport mechanism).

wait-status is **nowait** for all but "single-threaded" datagram servers — servers which do not release the socket until a timeout occurs (such as **comsat**(8C) and **talkd**(8C)). These must have the status **wait**. Although **tftpd**(8C) establishes separate "pseudo-connections", its forking behavior can lead to a race condition unless it is also given the status **wait**.

uid is the user ID under which the server should run. This allows servers to run with access privileges other than those for root.

server-program is either the pathname of a server program to be invoked by **inetd** to perform the requested service, or the value **internal** if **inetd** itself provides the service.

server-arguments If a server must be invoked with command-line arguments, the entire command line (including argument 0) must appear in this field (which consists of all remaining words in the entry). If the server expects **inetd** to pass it the address of its peer (for compatibility with 4.2BSD executable daemons), then the first argument to the command should be specified as '**%A**'.

FILES

/etc/inetd.conf
/etc/services
/etc/protocols

SEE ALSO

services(5), **comsat**(8C), **inetd**(8C), **talkd**(8C), **tftpd**(8C)

BUGS

inetd dumps core when the **inetd.conf** file contains blank lines.

NAME

internat – key mapping table for internationalization

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

This file format is used for the file specified by the –f option of **old-setkeys**(1).

The file has three columns. First column is keytable identifier, one of: BASE, CTRL, SHIFT, CAPS, UP, BASE_ISO, SHIFT_ISO or ALTG. The second column is a decimal keystation number. The third column is hexadecimal keytable entry value. The file must end with line of "END, 0, 0". As usual, comment lines start with #.

EXAMPLES

This is the file for mapping keys to Canadian standards:

```
# /usr/lib/.setkeys: Key remapping, used by "setkeys remap"
#
# First column is keytable identifier:
#                   BASE, CTRL, SHIFT, CAPS, UP, BASE_ISO, SHIFT_ISO or ALTG
# Second column is decimal keystation number
# Third column is hexadecimal keytable entry value
# File must end with line of "END, 0, 0"
# Comment lines must start with #
#
#
# --- Keymaps for Canadian keyboard ---
# > Define Alt Graph key (SHIFTKEYS+ALTGRAPH=86)
BASE 119 86
CTRL 119 86
SHIFT 119 86
CAPS 119 86
UP 119 86
# > Define Caps key (SHIFTKEYS+CAPSLOCK=80)
BASE 13 80
CTRL 13 80
SHIFT 13 80
CAPS 13 80
# > Define Floating Accent keys
#                   FA_UMLAUT = A9
#                   FA_CFLEX = AA
#                   FA_TILDE = AB
#                   FA_CEDILLA = AC
#                   FA_ACUTE = AD
#                   FA_GRAVE = AE
BASE 64 AA
SHIFT 64 A9
CAPS 64 A9
BASE 65 AC
SHIFT 65 AB
CAPS 65 AB
BASE 87 AE
SHIFT 87 AD
CAPS 87 AD
# > Define ASCII values
```

```
BASE 88 5B
SHIFT 88 7B
CAPS 88 7B
BASE 15 5D
SHIFT 15 7D
CAPS 15 7D
SHIFT 31 22
SHIFT 32 2F
SHIFT 35 3F
SHIFT 107 27
CAPS 107 27
SHIFT 108 60
CAPS 108 60
BASE 124 3C
SHIFT 124 3E
CAPS 124 3E
# > Define ISO values
BASE_ISO 109 E9
SHIFT_ISO 109 C9
# > Define Alternate Graph ISO values
ALTG 88 AB
ALTG 15 BB
ALTG 30 B1
ALTG 31 B2
ALTG 32 B3
ALTG 33 A2
ALTG 34 A4
ALTG 35 5E
ALTG 36 40
ALTG 37 A3
ALTG 38 5C
ALTG 40 AC
ALTG 41 23
ALTG 63 B6
ALTG 64 BC
ALTG 65 BD
ALTG 42 BE
ALTG 106 B5
ALTG 105 BA
# > End of file
END 0 0
```

SEE ALSO

old-setkeys(1)

The *Sun386i Developer's Guide* for keystation number diagrams.

NAME

ipalloc.netrange – range of addresses to allocate

SYNOPSIS

/etc/ipalloc.netrange

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

This file, if it exists on the Network Information Service (NIS) master of the **hosts.byaddr** map, specifies the ranges of IP addresses that can be allocated by the **ipallocd**(8C) daemon. This allows multiple address assignment authorities, probably in multiple administrative domains, to coexist on the same IP network by preallocating ranges of addresses. If this file does not exist, the daemon assumes that all addresses not listed in the **hosts** map may be freely allocated.

This file can contain blank lines. Comments begin with a '#' character and extend to the end of the current line. Ranges of free addresses are specified on one line per network or subnetwork.

The first token on the line is the IP address, in four part "dot" notation as also used in the **hosts** file, of the network or subnetwork described. It is separated from the second token by white space. The second token is a comma-separated list of local host number ranges on that network. These ranges take two forms: a single number specifies just that local host number, and two numbers separated by a dash specify all local host numbers starting at the first number and ending at the second. In the case of a subnet, host numbers not in that subnet are excluded.

For example, the following file would specify that a subset of the addresses on the class C network 192.9.200.0 may be allocated, and only some of the addresses on two particular subnets of the class B network 128.255.0.0 may be allocated. In any case, only non-broadcast addresses not listed in the **hosts** map are subject to allocation:

 # We have three network cables administered using automatic # IP address allocation.

192.9.200.0	**50-100,200-254**	**128.255.210.0**	**3,5,7,9,100-110**
128.255.211.0	**1-254**		

SEE ALSO

hosts(5), **netmasks**(5), **ipallocd**(8C)

BUGS

There is a silent limit of twenty ranges per network.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

keytables – keyboard table descriptions for loadkeys and dumpkeys

DESCRIPTION

These files are used by **loadkeys**(1) to modify the translation tables used by the keyboard streams module **kb**(4M), and generated by **dumpkeys** (see **loadkeys**(1)) from those translation tables.

Any line in the file beginning with # is a comment, and is ignored. # is treated specially only at the beginning of a line.

Other lines specify the values to load into the tables for a particular keystation. The format is either:

key *number list_of_entries*

or

swap *number1* **with** *number2*

or

key *number1* **same as** *number2*

or a blank line, which is ignored.

key *number list_of_entries*

sets the entries for keystation *number* from the list given. An entry in that list is of the form

tablename code

where *tablename* is the name of a particular translation table, or **all**. The translation tables are:

base	entry when no shifts are active
shift	entry when "Shift" key is down
caps	entry when "Caps Lock" is in effect
ctrl	entry when "Control" is down
altg	entry when "Alt Graph" is down
numl	entry when "Num Lock" is in effect
up	entry when a key goes up

All tables other than **up** refer to the action generated when a key goes down. Entries in the **up** table are used only for shift keys, since the shift in question goes away when the key goes up, except for keys such as "Caps Lock" or "Num Lock"; the keyboard streams module makes the key look as if it were a latching key.

A table name of **all** indicates that the entry for all tables should be set to the specified value, with the following exception: for entries with a value other than **hole**, the entry for the **numl** table should be set to **nonl**, and the entry for the **up** table should be set to **nop**.

The *code* specifies the effect of the key in question when the specified shift key is down. A *code* consists of either:

- A character, which indicates that the key should generate the given character. The character can either be a single character, a single character preceded by ^ which refers to a "control character" (for instance, ^**c** is control-C), or a C-style character constant enclosed in single quote characters ('), which can be expressed with C-style escape sequences such as \r for RETURN or \000 for the null character. Note that the single character may be any character in an 8-bit character set, such as ISO 8859/1.
- A string, consisting of a list of characters enclosed in double quote characters ("). Note that the use of the double quote character means that a *code* of double quote must be enclosed in single quotes.

- One of the following expressions:

shiftkeys+leftshift
the key is to be the left-hand "Shift" key

shiftkeys+rightshift
the key is to be the right-hand "Shift" key

shiftkeys+leftctrl
the key is to be the left-hand "Control" key

shiftkeys+rightctrl
the key is to be the right-hand "Control" key

shiftkeys+alt the key is to be the "Alt" shift key

shiftkeys+altgraph
the key is to be the "Alt Graph" shift key

shiftkeys+capslock
the key is to be the "Caps Lock" key

shiftkeys+shiftlock
the key is to be the "Shift Lock" key

shiftkeys+numlock
the key is to be the "Num Lock" key

buckybits+systembit
the key is to be the "Stop" key in Sunview; this is normally the L1 key, or the SETUP key on the VT100 keyboard

buckybits+metabit
the key is to be the "meta" key, that is, the "Left" or "Right" key on a Sun-2 or Sun-3 keyboard or the "diamond" key on a Sun-4 keyboard

compose the key is to be the "Compose" key

ctrlq on the "VT100" keyboard, the key is to transmit the control-Q character (this would be the entry for the "Q" key in the **ctrl** table)

ctrls on the "VT100" keyboard, the key is to transmit the control-S character (this would be the entry for the "S" key in the **ctrl** table)

noscroll on the "VT100" keyboard, the key is to be the "No Scroll" key

string+uparrow the key is to be the "up arrow" key

string+downarrow
the key is to be the "down arrow" key

string+leftarrow the key is to be the "left arrow" key

string+rightarrow
the key is to be the "right arrow" key

string+homearrow
the key is to be the "home" key

fa_acute the key is to be the acute accent "floating accent" key

fa_cedilla the key is to be the cedilla "floating accent" key

fa_cflex the key is to be the circumflex "floating accent" key

fa_grave the key is to be the grave accent "floating accent" key

fa_tilde the key is to be the tilde "floating accent" key

fa_umlaut	the key is to be the umlaut "floating accent" key
nonl	this is used only in the Num Lock table; the key is not to be affected by the state of Num Lock
pad0	the key is to be the "0" key on the numeric keypad
pad1	the key is to be the "1" key on the numeric keypad
pad2	the key is to be the "2" key on the numeric keypad
pad3	the key is to be the "3" key on the numeric keypad
pad4	the key is to be the "4" key on the numeric keypad
pad5	the key is to be the "5" key on the numeric keypad
pad6	the key is to be the "6" key on the numeric keypad
pad7	the key is to be the "7" key on the numeric keypad
pad8	the key is to be the "8" key on the numeric keypad
pad9	the key is to be the "9" key on the numeric keypad
paddot	the key is to be the "." key on the numeric keypad
padenter	the key is to be the "Enter" key on the numeric keypad
padplus	the key is to be the "+" key on the numeric keypad
padminus	the key is to be the "-" key on the numeric keypad
padstar	the key is to be the "*" key on the numeric keypad
padslash	the key is to be the "/" key on the numeric keypad
padequal	the key is to be the "=" key on the numeric keypad
padsep	the key is to be the "," (separator) key on the numeric keypad
lf(*n*)	the key is to be the left-hand function key *n*
rf(*n*)	the key is to be the right-hand function key *n*
tf(*n*)	the key is to be the top function key *n*
bf(*n*)	the key is to be the "bottom" function key *n*
nop	the key is to do nothing
error	this code indicates an internal error; to be used only for keystation 126, and must be used there
idle	this code indicates that the keyboard is idle (that is, has no keys down); to be used only for all entries other than the **numl** and **up** table entries for keystation 127, and must be used there
oops	this key exists, but its action is not defined; it has the same effect as **nop**
reset	this code indicates that the keyboard has just been reset; to be used only for the **up** table entry for keystation 127, and must be used there

swap *number1* **with** *number2*

exchanges the entries for keystations *number1* and *number2*.

key *number1* **same as** *number2*

sets the entries for keystation *number1* to be the same as those for keystation *number2*. If the file does not specify entries for keystation *number2*, the entries currently in the translation table are used; if the file does specify entries for keystation *number2*, those entries are used.

EXAMPLES

The following entry sets keystation 15 to be a "hole" (that is, an entry indicating that there is no keystation 15); sets keystation 30 to do nothing when Alt Graph is down, generate "!" when Shift is down, and generate "1" under all other circumstances; and sets keystation 76 to be the left-hand Control key.

```
key 15  all hole
key 30  base 1 shift ! caps 1 ctrl 1 altg nop
key 76  all shiftkeys+leftctrl up shiftkeys+leftctrl
```

The following entry exchanges the Delete and Back Space keys on the Type 4 keyboard:

```
swap 43 with 66
```

Keystation 43 is normally the Back Space key, and keystation 66 is normally the Delete key.

The following entry disables the Caps Lock key on the Type 3 and U.S. Type 4 keyboards:

```
key 119 all nop
```

The following specifies the standard translation tables for the U.S. Type 4 keyboard:

```
key 0   all hole
key 1   all buckybits+systembit up buckybits+systembit
key 2   all hole
key 3   all lf(2)
key 4   all hole
key 5   all tf(1)
key 6   all tf(2)
key 7   all tf(10)
key 8   all tf(3)
key 9   all tf(11)
key 10  all tf(4)
key 11  all tf(12)
key 12  all tf(5)
key 13  all shiftkeys+altgraph up shiftkeys+altgraph
key 14  all tf(6)
key 15  all hole
key 16  all tf(7)
key 17  all tf(8)
key 18  all tf(9)
key 19  all shiftkeys+alt up shiftkeys+alt
key 20  all hole
key 21  all rf(1)
key 22  all rf(2)
key 23  all rf(3)
key 24  all hole
key 25  all lf(3)
key 26  all lf(4)
key 27  all hole
key 28  all hole
key 29  all ^[
key 30  base 1 shift ! caps 1 ctrl 1 altg nop
key 31  base 2 shift @ caps 2 ctrl ^@ altg nop
key 32  base 3 shift # caps 3 ctrl 3 altg nop
key 33  base 4 shift $ caps 4 ctrl 4 altg nop
key 34  base 5 shift % caps 5 ctrl 5 altg nop
key 35  base 6 shift ^ caps 6 ctrl ^^ altg nop
key 36  base 7 shift & caps 7 ctrl 7 altg nop
```

```
key 37   base 8 shift * caps 8 ctrl 8 altg nop
key 38   base 9 shift ( caps 9 ctrl 9 altg nop
key 39   base 0 shift ) caps 0 ctrl 0 altg nop
key 40   base - shift _ caps - ctrl ^_ altg nop
key 41   base = shift + caps = ctrl = altg nop
key 42   base ` shift ~ caps ` ctrl ^^ altg nop
key 43   all '\b'
key 44   all hole
key 45   all rf(4) numl padequal
key 46   all rf(5) numl padslash
key 47   all rf(6) numl padstar
key 48   all bf(13)
key 49   all lf(5)
key 50   all bf(10) numl padequal
key 51   all lf(6)
key 52   all hole
key 53   all '\t'
key 54   base q shift Q caps Q ctrl ^Q altg nop
key 55   base w shift W caps W ctrl ^W altg nop
key 56   base e shift E caps E ctrl ^E altg nop
key 57   base r shift R caps R ctrl ^R altg nop
key 58   base t shift T caps T ctrl ^T altg nop
key 59   base y shift Y caps Y ctrl ^Y altg nop
key 60   base u shift U caps U ctrl ^U altg nop
key 61   base i shift I caps I ctrl '\t' altg nop
key 62   base o shift O caps O ctrl ^O altg nop
key 63   base p shift P caps P ctrl ^P altg nop
key 64   base [ shift { caps [ ctrl ^[ altg nop
key 65   base ] shift } caps ] ctrl ^] altg nop
key 66   all '\177'
key 67   all compose
key 68   all rf(7) numl pad7
key 69   all rf(8) numl pad8
key 70   all rf(9) numl pad9
key 71   all bf(15) numl padminus
key 72   all lf(7)
key 73   all lf(8)
key 74   all hole
key 75   all hole
key 76   all shiftkeys+leftctrl up shiftkeys+leftctrl
key 77   base a shift A caps A ctrl ^A altg nop
key 78   base s shift S caps S ctrl ^S altg nop
key 79   base d shift D caps D ctrl ^D altg nop
key 80   base f shift F caps F ctrl ^F altg nop
key 81   base g shift G caps G ctrl ^G altg nop
key 82   base h shift H caps H ctrl '\b' altg nop
key 83   base j shift J caps J ctrl '\n' altg nop
key 84   base k shift K caps K ctrl '\v' altg nop
key 85   base l shift L caps L ctrl ^L altg nop
key 86   base ; shift : caps ; ctrl ; altg nop
key 87   base '\'' shift '"' caps '\'' ctrl '\'' altg nop
key 88   base '\\' shift | caps '\\' ctrl ^\ altg nop
key 89   all '\r'
```

```
key 90  all bf(11) numl padenter
key 91  all rf(10) numl pad4
key 92  all rf(11) numl pad5
key 93  all rf(12) numl pad6
key 94  all bf(8) numl pad0
key 95  all lf(9)
key 96  all hole
key 97  all lf(10)
key 98  all shiftkeys+numlock
key 99  all shiftkeys+leftshift up shiftkeys+leftshift
key 100 base z shift Z caps Z ctrl ^Z altg nop
key 101 base x shift X caps X ctrl ^X altg nop
key 102 base c shift C caps C ctrl ^C altg nop
key 103 base v shift V caps V ctrl ^V altg nop
key 104 base b shift B caps B ctrl ^B altg nop
key 105 base n shift N caps N ctrl ^N altg nop
key 106 base m shift M caps M ctrl '\r' altg nop
key 107 base , shift < caps , ctrl , altg nop
key 108 base . shift > caps . ctrl . altg nop
key 109 base / shift ? caps / ctrl ^_ altg nop
key 110 all shiftkeys+rightshift up shiftkeys+rightshift
key 111 all '\n'
key 112 all rf(13) numl pad1
key 113 all rf(14) numl pad2
key 114 all rf(15) numl pad3
key 115 all hole
key 116 all hole
key 117 all hole
key 118 all lf(16)
key 119 all shiftkeys+capslock
key 120 all buckybits+metabit up buckybits+metabit
key 121 base ' ' shift ' ' caps ' ' ctrl ^@ altg ' '
key 122 all buckybits+metabit up buckybits+metabit
key 123 all hole
key 124 all hole
key 125 all bf(14) numl padplus
key 126 all error numl error up hole
key 127 all idle numl idle up reset
```

SEE ALSO

loadkeys(1), **kb**(4M)

NAME

link – link editor interfaces

SYNOPSIS

#include <link.h>

DESCRIPTION

Dynamically linked executables created by **ld**(1) contain data structures used by the dynamic link editor to finish link-editing the program during program execution. These data structures are described with a **link_dynamic** structure, as defined in the **link.h** file. **ld** always identifies the location of this structure in the executable file with the symbol **__DYNAMIC**. This symbol is **ld**-defined and if referenced in an executable that does not require dynamic linking will have the value zero.

The program stub linked with "main" programs by compiler drivers such as **cc**(1V) (called **crt0**) tests the definition of **__DYNAMIC** to determine whether or not the dynamic link editor should be invoked. Programs supplying a substitute for **crt0** must either duplicate this functionality or else require that the programs with which they are linked be linked *statically*. Otherwise, such replacement **crt0**'s must open and map in the executable **/usr/lib/ld.so** using **mmap**(2). Care should be taken to ensure that the expected mapping relationship between the "text" and "data" segments of the executable is maintained in the same manner that the **execve**(2V) system call does. The first location following the **a.out** header of this executable is the entry point to a function that begins the dynamic link-editing process. This function must be called and supplied with two arguments. The first argument is an integer representing the revision level of the argument list, and should have the value "1". The second should be a pointer to an argument list structure of the form:

```
struct {
        int         crt_ba;          /* base address of ld.so */
        int         crt_dzfd;        /* open fd to /dev/zero */
        int         crt_ldfd;        /* open fd to ld.so */
        struct      link_dynamic *crt_dp; /* pointer to program's __DYNAMIC */
        char        **crt_ep;        /* environment strings */
        caddr_t     crt_bp;          /* debugger hook */
}
```

The members of the structure are:

crt_ba The address at which **/usr/lib/ld.so** has been mapped.

crt_dzfd An open file descriptor for **/dev/zero**. **ld.so** will close this file descriptor before returning.

crt_ldfd The file descriptor used to map **/usr/lib/ld.so**. **ld.so** will close this file descriptor before returning.

crt_dp A pointer to the label **__DYNAMIC** in the executable which is calling **ld.so**.

crt_ep A pointer to the environment strings provided to the program.

crt_bp A location in the executable which contains an instruction that will be executed after the call to **ld.so** returns. This location is used as a breakpoint in programs that are being executed under the control of a debugger such as **adb**(1).

SEE ALSO

ld(1), **mmap**(2), **a.out**(5)

BUGS

These interfaces are under development and are subject to rapid change.

NAME

locale – locale database

SYNOPSIS

/usr/share/lib/locale/*category/locale*

/etc/locale/*category/locale*

DESCRIPTION

The *category* directory contains information relating to one category of the complete list of categories that comprise a full locale for all systems sharing this directory. *locale* is either a file or a directory that contains information relating to the relevant category indicated by its parent directory *category*. *locale* is the name that is given to describe the style of operation required by an application in a particular language, territory or code-set.

At runtime these directories will be accessed if the application has made a valid call to:

setlocale(*category, locale*)

where *category* can be any one of the following settings:

LC_COLLATE Collation order. Affects the behavior of regular expressions and the string functions defined in **strcoll**(3).

LC_CTYPE Character classification and case conversion. Affects the behavior of regular expressions and the character handling functions defined in **toascii**(3), and **ctime**(3V).

LC_MONETARY Monetary formatting. Affects the behavior of functions that handle monetary values.

LC_NUMERIC Numeric delimiters. Affects the radix character of the formatted input/output functions defined in **printf**(3V) and **scanf**(3V), and the conversion functions defined in **strtod**(3).

LC_TIME Date and time formats. Affects the behavior of the time functions defined in **ctime**(3V).

LC_MESSAGES Message presentation style. Affects the behavior of the string access functions defined in **catgets**(3C) and **gettext**(3).

NLSPATH Contains a sequence of pseudo-pathnames which **catopen**(3C) uses when attempting to locate message catalogs. Each pseudo-pathname contains a name template consisting of an optional path-prefix, one or more substitution fields, a filename and an optional filename suffix.

Substitution fields consist of a % symbol, followed by a single-letter keyword. The following keywords are currently defined:

%N The value of the *name* parameter passed to **catopen**(3C).

%L The value of the **LANG** environment variable.

%% A single % character.

A null string is sustituted if the specified value is not defined. Pathnames defined in **NLSPATH** are separated by colons (**:**). A leading or two adjacent colons indicate the current directory. For example:

NLSPATH=":%N.cat:/nlslib/%L/%N.cat"

Indicates to **catopen**(3C) that it should look for the requested message catalog in *name, name.cat* and */nlslib/***$LANG**/*name.cat*. The **LC_ALL** and **LANG** environment variables do not commute to real directories or files but instead relate to a locale that is a assumed to be valid for all of the above categories.

SEE ALSO

catgets(3C), **catopen**(3C), **ctime**(3V), **gettext**(3), **printf**(3V), **scanf**(3V), **setlocale**(3V), **strcoll**(3) **strtod**(3), **toascii**(3V)

NAME

magic – file command's magic number file

DESCRIPTION

The **file**(1) command identifies the type of a file using, among other tests, a test for whether the file begins with a certain *magic number*. The file **/etc/magic** specifies what magic numbers are to be tested for, what message to print if a particular magic number is found, and additional information to extract from the file.

Each line of the file specifies a test to be performed. A test compares the data starting at a particular offset in the file with a 1-byte, 2-byte, or 4-byte numeric value or a string. If the test succeeds, a message is printed. The line consists of the following fields:

offset *type* *value* *message*

offset A number specifying the offset, in bytes, into the file of the data which is to be tested.

type The type of the data to be tested. The possible values are:

byte A one-byte value.

short A two-byte value.

long A four-byte value.

string A string of bytes.

The types **byte**, **short**, and **long** may optionally be followed by a mask specifier of the form **&***number*. If a mask specifier is given, the value is AND'ed with the *number* before any comparisons are done. The *number* is specified in C form. For instance, **13** is decimal, **013** is octal, and **0x13** is hexadecimal.

value The value to be compared with the value from the file. If the type is numeric, this value is specified in C form. If it is a string, it is specified as a C string with the usual escapes permitted (for instance, \n for NEWLINE).

Numeric values may be preceded by a character indicating the operation to be performed. It may be '=', to specify that the value from the file must equal the specified value, '<', to specify that the value from the file must be less than the specified value, '>', to specify that the value from the file must be greater than the specified value, '&', to specify that all the bits in the specified value must be set in the value from the file, '^', to specify that at least one of the bits in the specified value must not be set in the value from the file, or x to specify that any value will match. If the character is omitted, it is assumed to be '='.

For string values, the byte string from the file must match the specified byte string. The byte string from the file which is matched is the same length as the specified byte string.

message The message to be printed if the comparison succeeds. If the string contains a **printf**(3V) format specification, the value from the file (with any specified masking performed) is printed using the message as the format string.

Some file formats contain additional information which is to be printed along with the file type. A line which begins with the character '>' indicates additional tests and messages to be printed. If the test on the line preceding the first line with a '>' succeeds, the tests specified in all the subsequent lines beginning with '>' are performed, and the messages printed if the tests succeed. The next line which does not begin with a '>' terminates this.

FILES

/etc/magic

SEE ALSO

file(1), **printf**(3V)

BUGS

There should be more than one level of subtests, with the level indicated by the number of '>' at the beginning of the line.

NAME

mtab – mounted file system table

SYNOPSIS

/etc/mtab

#include <mntent.h>

DESCRIPTION

mtab resides in the **/etc** directory, and contains a table of filesystems currently mounted by the **mount**(8) command. **umount** removes entries from this file.

The file contains a line of information for each mounted filesystem, structurally identical to the contents of **/etc/fstab**, described in **fstab**(5). There are a number of lines of the form:

fsname dir type opts freq passno

for example:

/dev/xy0a / 4.2 rw,noquota 1 2

The file is accessed by programs using **getmntent**(3), and by the system administrator using a text editor.

FILES

/etc/mtab
/etc/fstab

SEE ALSO

getmntent(3), **fstab**(5), **mount**(8)

NAME

netgroup – list of network groups

DESCRIPTION

netgroup defines network wide groups, used for permission checking when doing remote mounts, remote logins, and remote shells. For remote mounts, the information in **netgroup** is used to classify machines; for remote logins and remote shells, it is used to classify users. Each line of the **netgroup** file defines a group and has the format

groupname list-of-members

where members is either another group name, or a triple:

(*hostname*, *username*, *domainname*)

Any of these three fields can be empty, in which case it signifies a wild card. Thus

universal (,,)

defines a group to which everyone belongs.

The *domainname* field must either be the local domain name or empty for the netgroup entry to be used. This field does *not* limit the netgroup or provide security. The *domainname* field refers to the domain in which the triple is valid, not the domain containing the trusted host.

A gateway machine should be listed under all possible hostnames by which it may be recognized:

wan (gateway,,) (gateway-ebb,,)

Field names that begin with something other than a letter, digit or underscore (such as '–') work in precisely the opposite fashion. For example, consider the following entries:

justmachines (analytica,–,sun)
justpeople (–,babbage,sun)

The machine **analytica** belongs to the group **justmachines** in the domain **sun**, but no users belong to it. Similarly, the user **babbage** belongs to the group **justpeople** in the domain **sun**, but no machines belong to it.

SEE ALSO

getnetgrent(3N), **exports**(5), **makedbm**(8), **ypserv**(8)

WARNINGS

The triple, (,,*domain*), allows all users and machines trusted access, and has the same effect as the triple, (,,).

To correctly restrict access to a specific set of members, use the *hostname* and *username* fields of the triple.

NAME

netmasks – network mask data base

DESCRIPTION

The **netmasks** file contains network masks used to implement IP standard subnetting. For each network that is subnetted, a single line should exist in this file with the network number, any number of SPACE or TAB characters, and the network mask to use on that network. Network numbers and masks may be specified in the conventional IP '.' notation (like IP host addresses, but with zeroes for the host part). For example,

128.32.0.0 255.255.255.0

can be used to specify the Class B network 128.32.0.0 should have eight bits of subnet field and eight bits of host field, in addition to the standard sixteen bits in the network field. When running the Network Information Service (NIS), this file on the master is used for the **netmasks.byaddr** map.

FILES

/etc/netmasks

SEE ALSO

ifconfig(8C)

Postel, Jon, and Mogul, Jeff, *Internet Standard Subnetting Procedure*, RFC 950, Network Information Center, SRI International, Menlo Park, Calif., August 1985.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

netrc – file for ftp remote login data

DESCRIPTION

The **.netrc** file contains data for logging in to a remote host over the network for file transfers by **ftp**(1C). This file resides in the user's home directory on the machine initiating the file transfer. Its permissions should be set to disallow read access by group and others (see **chmod**(1V)).

The following tokens are recognized; they may be separated by SPACE, TAB, or NEWLINE characters:

machine*name*

Identify a remote machine name. The auto-login process searches the **.netrc** file for a **machine** token that matches the remote machine specified on the **ftp** command line or as an **open** command argument. Once a match is made, the subsequent **.netrc** tokens are processed, stopping when the EOF is reached or another **machine** token is encountered.

login *name*

Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified name.

password *string*

Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note: if this token is present in the **.netrc** file, **ftp** will abort the auto-login process if the **.netrc** is readable by anyone besides the user.

account *string*

Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an **ACCT** command if it does not.

macdef *name*

Define a macro. This token functions as the **ftp macdef** command functions. A macro is defined with the specified name; its contents begin with the next **.netrc** line and continue until a null line (consecutive NEWLINE characters) is encountered. If a macro named **init** is defined, it is automatically executed as the last step in the auto-login process.

EXAMPLE

The command:

machine ray login demo password mypassword

allows an autologin to the machine **ray** using the login name **demo** with password **mypassword**.

FILES

~/.netrc

SEE ALSO

chmod(1V), **ftp**(1C), **ftpd**(8C)

NAME

networks – network name data base

DESCRIPTION

The **networks** file contains information regarding the known networks which comprise the TCP/IP. For each network a single line should be present with the following information:

official-network-name network-number aliases

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official network data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown networks.

Network number may be specified in the conventional '.' notation using the **inet_network** ( ) routine from the Internet address manipulation library, **inet**(3N). Network names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

FILES

/etc/networks

SEE ALSO

getnetent(3N), **inet**(3N)

BUGS

A name server should be used instead of a static file. A binary indexed file format should be available for fast access.

NAME

orgrc – organizer configuration and initialization file

AVAILABILITY

Sun386i systems only.

DESCRIPTION

organizer(1) is a SunView 1 application for viewing and manipulating files and directories. It saves its parameters in the **.orgrc** file between runs. The user can use this file to configure **organizer**.

The first parameter in the file should always be the version number.

Version = 1.1

Change the version number only when necessary; if **organizer** determines that this version is "old", then it will save this version in **~/.orgrc.old** and try to copy **/usr/lib/Orgrc** into **~/.orgrc**.

The next two parameters assign default names for the system DOS Program and the default text editor.

DOS Program = dos
Text Editor = textedit

The DOS Program parameter should not be changed. However, the user can change the default text editor. For example:

Text Editor = shelltool vi

The Properties section initializes or customizes certain properties. The possible values for each item are listed below. The braces and vertical bars below indicate choices, they are not used in the **.orgrc** file. The **Update Interval** is in seconds.

 # Properties
PROPERTY Display Style = {Name and Icon | Name Only | Name and Info}
PROPERTY Roadmap = {Yes | No}
PROPERTY Show Hidden Files = {Yes | No}
PROPERTY Sort Type = {Name | File Type | Size | Date}
PROPERTY Sort Direction = {Ascending | Descending}
PROPERTY Update Interval = [5-300]

The Color Palette specifies all the color values used by **organizer**'s buttons and icons. These values must be RGB triplets. It is listed below.

Begin Color Palette
Background Color = 255, 255, 255
Directory Name Color = 0, 146, 236
Directory Icon Foreground Color = 114, 45, 0
Directory Icon Background Color = 255, 227, 185
Directory Highlight Name Color = 255, 255, 255
Text Name Color = 0, 166, 143
Text Icon Foreground Color = 0, 0, 0
Text Icon Background Color = 255, 255, 255
Text Highlight Name Color = 255, 255, 255
Executable Name Color = 255, 0, 0
Executable Icon Foreground Color = 157, 162, 187
Executable Icon Background Color = 255, 255, 255
Executable Highlight Name Color = 255, 255, 255
Device Name Color = 113, 117, 135
Device Icon Foreground Color = 0, 0, 0
Device Icon Background Color = 174, 255, 159
Device Highlight Name Color = 255, 255, 255
Button Group1 Color = 255, 220, 187

```
    Button Group2 Color = 201, 211, 232
    Button Group3 Color = 255, 244, 113
    Button Foreground Color = 0, 0, 0
    Button Background Color = 255, 255, 255
    Button Shadow Color = 180, 180, 184
    Button Highlight Color = 0, 0, 0
    Scrollbar Color = 142, 106, 146
  End Color Palette
```

The Color Labels section allows the labelling or "aliasing" of RGB triplets. The right side of a label assignment can contain an RGB triplet, a palette entry, or another label that has already been assigned. Here's an example:

```
  Begin Color Labels
    Black = Text Icon Foreground Color
    White = Background Color
    Orange = 255, 213, 127
    Dark Red = 232, 0, 0
    Steel Blue = 114, 146, 161
    Rasberry (sic) = 202, 140, 156
    Dark Blue = 0, 75, 161
    Light Gray = 223, 223, 223
    Maroon = 182, 84, 106
  End Color Labels
```

The rest of the **.orgrc** file contains user defined file types. The user can specify that certain files be grouped together and treated in a similar fashion. That is, the same icon is used to display all files in a file type, and the same command is used when a file is opened or edited. In the default **.orgrc** (**/usr/lib/Orgrc**) there are ten user defined file types. Here is an example of a user defined file type:

```
  Begin File Type Definition
    Name = *.c
    Background Icon = /usr/include/images/cMask.icon
    Foreground Icon = /usr/include/images/cStencil.icon
    Name Color = Black
    Icon Background Color = Orange
    Icon Foreground Color = Black
    Highlight Name Color = White
    Execute Application = cmdtool vi "$(FILE)"
    Edit Application = cmdtool vi "$(FILE)"
    Print Application = pr -f "$(FILE)" | lpr
  End File Type Definition
```

The right side of the **Name** field can contain any combination of **csh**(1) **Filename Substitution** characters. This field specifies the file type by way of its name. The next six fields together specify an **organizer** icon. This model allows a rich variety of icons. For more information, see the *Sun386i Advanced Skills* manual. The right side of the **Execute Application** entry specifies the command to execute when the user either opens or double clicks on a file of that type. The **Edit Application** and **Print Application** entries specify the command to execute when the user requests that a file of that type be edited or printed.

FILES

~/.orgrc	read at beginning of execution by the Organizer
/usr/lib/Orgrc	default **.orgrc** file

SEE ALSO

organizer(1)

Sun386i User's Guide
Sun386i Advanced Skills

LIMITATIONS

The right side of Color Palette entries must be RGB triplets.

Forward references for Color Labels are not allowed.

BUGS

organizer saves its parameters as it exits; unfortunately, it does not know how to save user's comments in the file. So, comments are blown away.

NAME

passwd – password file

SYNOPSIS

/etc/passwd

DESCRIPTION

The **passwd** file contains basic information about each user's account. This file contains a one-line entry for each authorized user, of the form:

username:*password*:*uid*:*gid*:*gcos-field*:*home-dir*:*login-shell*

where

username is the user's login name. This field contains no uppercase characters, and must not be more than eight characters in length.

password is the user's encrypted password, or a string of the form: *##name* if the encrypted password is in the **/etc/security/passwd.adjunct** file (see **passwd.adjunct**(5)). If this field is empty, **login**(1) does not request a password before logging the user in.

uid is the user's numerical ID for the system, which must be unique. *uid* is generally a value between 0 and 32767.

gid is the numerical ID of the group that the user belongs to. *gid* is generally a value between 0 an 32767.

gcos-field is the user's real name, along with information to pass along in a mail-message heading. It is called the gcos-field for historical reasons. A & in this field stands for the login name (in cases where the login name appears in a user's real name).

home-dir is the pathname to the directory in which the user is initially positioned upon logging in.

login-shell is the user's initial shell program. If this field is empty, the default shell is **/usr/bin/sh**.

The **passwd** file can also have lines beginning with a '+' (plus sign) which means to incorporate entries from the Network Information Service (NIS). There are three styles of + entries in this file: by itself, + means to insert the entire contents of the NIS password file at that point; *+name* means to insert the entry (if any) for *name* from the NIS service at that point; *+@netgroup* means to insert the entries for all members of the network group **netgroup** at that point. If a *+name* entry has a non-null *password*, *gcos*, *home-dir*, or *login-shell* field, the value of that field overrides what is contained in the NIS service. The *uid* and *gid* fields cannot be overridden.

The **passwd** file can also have lines beginning with a '–' (minus sign) which means to disallow entries from the NIS service. There are two styles of '–' entries in this file: *–name* means to disallow any subsequent entries (if any) for *name* (in this file or in the NIS service); *–@netgroup* means to disallow any subsequent entries for all members of the network group *netgroup*.

The password file is an ASCII file that resides in the **/etc** directory. Because the encrypted passwords on a secure system are kept in the **passwd.adjunct** file, **/etc/passwd** has general read permission on all systems, and can be used by routines that map numerical user IDs to names.

Appropriate precautions must be taken to lock the **/etc/passwd** file against simultaneous changes if it is to be edited with a text editor; **vipw**(8) does the necessary locking.

EXAMPLE

Here is a sample **passwd** file when **passwd.adjunct** does not exist:

```
root:q.mJzTnu8icF.:0:10:God:/:/bin/csh
fred:6k/7KCFRPNVXg:508:10:% Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+::::Guest
```

Here is a sample **passwd** file when **passwd.adjunct** does exist:

```
root:##root:0:10:God:/:/bin/csh
fred:##fred:508:10:& Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+::::Guest
```

In this example, there are specific entries for users **root** and **fred**, to assure that they can log in even when the system is running standalone. The user **john** will have his password entry in the NIS service incorporated without change; anyone in the netgroup **documentation** will have their password field disabled, and anyone else will be able to log in with their usual password, shell, and home directory, but with a gcos-field of **Guest.**

FILES

/etc/passwd
/etc/security/passwd.adjunct

SEE ALSO

login(1), **mail**(1), **passwd**(1), **crypt**(3), **getpwent**(3V), **group**(5), **passwd.adjunct**(5), **adduser**(8), **sendmail**(8), **vipw**(8)

BUGS

mail(1) and **sendmail**(8) use the gcos-field to compose the **From:** line for addressing mail messages, but these programs get confused by nested parentheses when composing replies. This problem can be avoided by using different types of brackets within the gcos-field; for example:

(& Fredricks [Podunk U <EE/CIS>] {818}-555-5555)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

passwd.adjunct – user security data file

SYNOPSIS

/etc/security/passwd.adjunct

DESCRIPTION

The **passwd.adjunct** file contains the following information for each user:

name:*password*:*min-label*:*max-label*:*default-label*:*always-audit-flags*:*never-audit-flags*:

name	The user's login name in the system and it must be unique.
password	The encrypted password.
min-label	The lowest security level at which this user is allowed to login (not used at C2 level).
max-label	The highest security level at which this user is allowed to login (not used at C2 level).
default-label	The security level at which this user will run unless a label is specified at login.
always-audit-flags	Flags specifying events always to be audited for this user's processes; see **audit_control**(5).
never-audit-flags	Flags specifying events never to be audited for this user's processes; see **audit_control**(5).

Field are separated by a colon, and each user from the next by a NEWLINE.

The **passwd.adjunct** file can also have line beginning with a '+' (plus sign), which means to incorporate entries from the Network Information Service (NIS). There are three styles of '+' entries: all by itself, '+' means to insert the entire contents of the NIS **passwd.adjunct** file at that point; *+name* means to insert the entry (if any) for *name* from the NIS service at that point; *+@name* means to insert the entries for all members of the network group *name* at that point. If a '+' entry has a non-null password, it will override what is contained in the NIS service.

EXAMPLE

Here is a sample **/etc/security/passwd.adjunct** file:

```
root:q.mJzTnu8icF.::::::
ignatz:7KsI8CFRPNVXg::b,ap,bp,gp,dp,ic,r,d,l::+dc,+da:-dr:
rex:7HU8UUGRPNVXg:b,ap:b,ap,bp:b,bp::+ad:
+fred:9x.FFUw6xcJBa::::::
+:
```

The user **root** is the super-user, who has no special label constraints nor audit interest. The user **ignatz** may have any label from the lowest to the level **b** and any of a large number of categories. **ignatz** will run at system low unless he specifies otherwise. He is being audited on the system default event classes as well as data creations and access changes, but never for failed data reads. The user **rex** can function only at the level **b** and only in the categories **ap** or **ap** and **bp**. By default, he will run at '**b,bp**'. He is audited with the system defaults, except that successful administrative operations are not audited. The user **fred** will have the labels and audit flags that are specified in the NIS **passwd.adjunct** file. Any other users specified in the NIS service will be able to log in on this system.

The user security data file resides in the **/etc/security** directory. Because it contains encrypted passwords, it does not have general read permission.

FILES

/etc/security/passwd.adjunct
/etc/security

SEE ALSO

login(1), **passwd**(1), **crypt**(3), **getpwaent**(3), **getpwent**(3V), **audit_control**(5), **passwd**(5), **adduser**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

phones – remote host phone number data base

SYNOPSIS

/etc/phones

DESCRIPTION

The file **/etc/phones** contains the system-wide private phone numbers for the **tip**(1C) program. **/etc/phones** is normally unreadable, and so may contain privileged information. The format of **/etc/phones** is a series of lines of the form:

<system-name>[\t]*****<*phone-number*>.

The system name is one of those defined in the **remote**(5) file and the phone number is constructed from **[0123456789**–=*****%]. The '=' and '*' characters are indicators to the auto call units to pause and wait for a second dial tone (when going through an exchange). The '=' is required by the DF02-AC and the '*' is required by the BIZCOMP 1030.

Comment lines are lines containing a '#' sign in the first column of the line.

Only one phone number per line is permitted. However, if more than one line in the file contains the same system name **tip**(1C) will attempt to dial each one in turn, until it establishes a connection.

FILES

/etc/phones

SEE ALSO

tip(1C), **remote**(5)

NAME

plot – graphics interface

DESCRIPTION

Files of this format are produced by routines described in **plot**(3X), and are interpreted for various devices by commands described in **plot**(1G). A graphics file is a stream of plotting instructions. Each instruction consists of an ASCII letter usually followed by bytes of binary information. The instructions are executed in order. A point is designated by four bytes representing the *x* and *y* values; each value is a signed integer. The last designated point in an *l*, *m*, *n*, or *p* instruction becomes the "current point" for the next instruction.

Each of the following descriptions begins with the name of the corresponding routine in **plot**(3X).

m Move: the next four bytes give a new current point.

n Cont: draw a line from the current point to the point given by the next four bytes. See **plot**(1G).

p Point: plot the point given by the next four bytes.

l Line: draw a line from the point given by the next four bytes to the point given by the following four bytes.

t Label: place the following ASCII string so that its first character falls on the current point. The string is terminated by a NEWLINE.

a Arc: the first four bytes give the center, the next four give the starting point, and the last four give the end point of a circular arc. The least significant coordinate of the end point is used only to determine the quadrant. The arc is drawn counter-clockwise.

c Circle: the first four bytes give the center of the circle, the next two the radius.

e Erase: start another frame of output.

f Linemod: take the following string, up to a NEWLINE, as the style for drawing further lines. The styles are "dotted," "solid," "longdashed," "shortdashed," and "dotdashed." Effective only in **plot 4014** and **plot ver**.

s Space: the next four bytes give the lower left corner of the plotting area; the following four give the upper right corner. The plot will be magnified or reduced to fit the device as closely as possible.

Space settings that exactly fill the plotting area with unity scaling appear below for devices supported by the filters of **plot**(1G). The upper limit is just outside the plotting area. In every case the plotting area is taken to be square; points outside may be displayable on devices whose face is not square.

4014	**space(0, 0, 3120, 3120);**
ver	**space(0, 0, 2048, 2048);**
300, 300s	**space(0, 0, 4096, 4096);**
450	**space(0, 0, 4096, 4096);**

SEE ALSO

graph(1G), **plot**(1G), **plot**(3X)

NAME

pnp.sysnames – file used to allocate system names

SYNOPSIS

/etc/pnp.sysnames

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **/etc/pnp.sysnames** file contains system names that may be allocated on demand, typically as part of Automatic System Installation.

The system names should be legal system names, one per line. Legal names are up to 31 characters long, and consist of lowercase alphanumeric characters, dashes, and underscores. The first character must be alphabetic, and the last character should be alphanumeric. Blank lines are allowed in the file, but comments are not.

When a system name needs to be allocated, the first unused system name is taken from **/etc/pnp.sysnames**. If all the system names there are in use, unused names are allocated from the list *system-1*, *system-2*, ...; the default prefix *system* may be changed in the **/var/yp/updaters** makefile. A system name is "used" if there is already a matching entry in the Network Information Service (NIS) *hosts.byname* map, the *ethers.byname* map, or there is a netgroup with that name. Names are allocated to correspond to a given Ethernet address. There is no concept of "transient" name allocation; part of allocating a system name includes updating the *ethers.byname* and *ethers.byaddr* NIS maps to persistently associate the name with that Ethernet address.

One way to allocate a system name is to issue a **ypupdate**(3N) call to update the *ethers.byaddr* map. The key is the Ethernet address (or general IEEE 802.2 48 bit address, used also with FDDI and Token Ring standards) of the system whose name is being allocated. The data is a line formatted according to the format specified in **ethers**(5). A name is allocated if the name passed is '*' (a single asterisk). Updating this NIS map using **ypupdate**(3N) is a privileged operation, and may be performed only by users in the *networks* group (with group ID 12), or boot servers (listed in the *ypservers* NIS map).

FILES

/etc/pnp.sysnames
/usr/etc/yp/upd.systems
/var/yp/updaters

SEE ALSO

ypupdate(3N), **ethers**(5), **group**(5), **hosts**(5), **netgroup**(5), **updaters**(5), **pnpd**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

policies – network administration policies

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **policies** file contains information relevant to domain-wide administration policies. Each line contains two tokens, separated by white space; the first token is the name of an administrative policy, and the second is the value of that policy.

FILES

/etc/policies
/var/yp/*domainname***/policies.{dir,pag}**

SEE ALSO

pnpd(8C), **rarpd**(8C), **logintool**(8)

NAME

printcap – printer capability data base

SYNOPSIS

/etc/printcap

DESCRIPTION

printcap is a simplified version of the **termcap**(5) data base for describing printers. The spooling system accesses the **printcap** file every time it is used, allowing dynamic addition and deletion of printers. Each entry in the data base describes one printer. This data base may not be substituted for, as is possible for **termcap**, because it may allow accounting to be bypassed.

The default printer is normally **lp**, though the environment variable PRINTER may be used to override this. Each spooling utility supports a **–P***printer* option to explicitly name a destination printer.

Refer to *System and Network Administration* for a discussion of how to set up the database for a given printer. On Sun386i systems, refer to **snap**(1) for information on setting up printers with the system and network administration program.

Each entry in the **printcap** file describes a printer, and is a line consisting of a number of fields separated by '**:**' characters. The first entry for each printer gives the names which are known for the printer, separated by '|' characters. The first name is conventionally a number. The second name given is the most common abbreviation for the printer, and the last name given should be a long name fully identifying the printer. The second name should contain no blanks; the last name may well contain blanks for readability. Entries may continue onto multiple lines by giving a '\' as the last character of a line, and empty fields may be included for readability.

Capabilities in **printcap** are all introduced by two-character codes, and are of three types:

Boolean Capabilities that indicate that the printer has some particular feature. Boolean capabilities are simply written between the '**:**' characters, and are indicated by the word 'bool' in the **type** column of the capabilities table below.

Numeric Capabilities that supply information such as baud-rates, number of lines per page, and so on. Numeric capabilities are indicated by the word **num** in the **type** column of the capabilities table below. Numeric capabilities are given by the two-character capability code followed by the '**#**' character, followed by the numeric value. The following example is a numeric entry stating that this printer should run at 1200 baud:

:br#1200:

String Capabilities that give a sequence which can be used to perform particular printer operations such as cursor motion. String valued capabilities are indicated by the word **str** in the **type** column of the capabilities table below. String valued capabilities are given by the two-character capability code followed by an '**=**' sign and then a string ending at the next following '**:**'. For example,

:rp=spinwriter:

is a sample entry stating that the remote printer is named **spinwriter**.

Sun386i DESCRIPTION

On Sun386i systems, **lpr**(1) and related printing commands use the Network Information Service (NIS) to obtain the **printcap** entry for a named printer if the entry does not exist in the local **/etc/printcap** file. For example, when a user issues the command:

lpr -Pnewprinter foo

lpr searches **/etc/printcap** on the local system for an entry for **newprinter**. If no local entry for **newprinter** exists, then **lpr** searches the NIS map called **printcap**. The search is invisible to the user.

lpr creates the spooling directory for the printer automatically if no spooling directory exists.

System administrators can make a printer available to the entire NIS domain by placing an entry for that printer in the NIS **printcap** map, typically using **snap**. Otherwise, the system administrator must edit the **/etc/printcap** file on the NIS master and then rebuild the NIS map.

CAPABILITIES

Name	*Type*	*Default*	*Description*
af	str	NULL	name of accounting file
br	num	none	if lp is a tty, set the baud rate (ioctl call)
cf	str	NULL	cifplot data filter
df	str	NULL	TeX data filter (DVI format)
du	str	0	User ID of user 'daemon'.
fc	num	0	if lp is a tty, clear flag bits
ff	str	"\f"	string to send for a form feed
fo	bool	false	print a form feed when device is opened
fs	num	0	like 'fc' but set bits
gf	str	NULL	graph data filter (plot(3X) format)
hl	bool	false	print the burst header page last
ic	bool	false	driver supports (non standard) ioctl to indent printout
if	str	NULL	name of input/communication filter (created per job)
lf	str	"/dev/console"	error logging file name
lo	str	"lock"	name of lock file
lp	str	"/dev/lp"	device name to open for output
mc	num	0	maximum number of copies
ms	str	NULL	list of terminal modes to set or clear
mx	num	1000	maximum file size (in BUFSIZ blocks), zero = unlimited
nd	str	NULL	next directory for list of queues (unimplemented)
nf	str	NULL	ditroff data filter (device independent troff)
of	str	NULL	name of output/banner filter (created once)
pc	num	200	price per foot or page in hundredths of cents
pl	num	66	page length (in lines)
pw	num	132	page width (in characters)
px	num	0	page width in pixels (horizontal)
py	num	0	page length in pixels (vertical)
rf	str	NULL	filter for printing FORTRAN style text files
rg	str	NULL	restricted group. Only members of group allowed access
rm	str	NULL	machine name for remote printer
rp	str	"lp"	remote printer name argument
rs	bool	false	restrict remote users to those with local accounts
rw	bool	false	open printer device read/write instead of write-only
sb	bool	false	short banner (one line only)
sc	bool	false	suppress multiple copies
sd	str	"/var/spool/lpd"	spool directory
sf	bool	false	suppress form feeds
sh	bool	false	suppress printing of burst page header
st	str	"status"	status file name
tc	str	NULL	name of similar printer; must be last
tf	str	NULL	troff data filter (C/A/T phototypesetter)
tr	str	NULL	trailer string to print when queue empties
vf	str	NULL	raster image filter
xc	num	0	if lp is a tty, clear local mode bits
xs	num	0	like 'xc' but set bits

If the local line printer driver supports indentation, the daemon must understand how to invoke it.

Note: the **fs**, **fc**, **xs**, and **xc** fields are flag *masks* rather than flag *values*. Certain default device flags are set when the device is opened by the line printer daemon if the device is connected to a terminal port. The flags indicated in the **fc** field are then cleared; the flags in the **fs** field are then set (or vice-versa, depending on the order of **fc***#nnnn* and **fs***#nnnn* in the **/etc/printcap** file). The bits cleared by the **fc** field and set by the **fs** field are those in the **sg_flags** field of the **sgtty** structure, as set by the **TIOCSETP ioctl** call, and the bits cleared by the **xc** field and set by the **xs** field are those in the "local flags" word, as set by the **TIOCLSET ioctl** call. See **ttcompat**(4M) for a description of these flags. For example, to set exactly the flags 06300 in the **fs** field, which specifies that the **EVENP**, **ODDP**, and **XTABS** modes are to be set, and all other flags are to be cleared, do:

:fc#0177777:fs#06300:

The same process applies to the **xc** and **xs** fields. Alternatively, the **ms** field can be used to specify modes to be set and cleared. These modes are specified as **stty**(1V) modes; any mode supported by **stty** may be specified, except for the baud rate which must be specified with the **br** field. This permits modes not supported by the older terminal interface described in **ttcompat**(4M) to be set or cleared. Thus, to set the terminal port to which the printer is attached to even parity, TAB expansion, no NEWLINE to RETURN/LINEFEED translation, and RTS/CTS flow control enabled, do:

:ms=evenp,-tabs,nl,crtscts:

On Sun386i systems, the **tc** field, as in the **termcap**(5) file, must appear last in the list of capabilities. It is recommended that each type of printer have a general entry describing common capabilities; then an individual printer can be defined with its particular capabilities plus a **tc** field that points to the general entry for that type of printer.

FILES

/etc/printcap

SEE ALSO

lpq(1), **lpr**(1), **lprm**(1), **plot**(1G), **snap**(1), **stty**(1V), **plot**(3X), **ttcompat**(4M), **termcap**(5), **lpc**(8), **lpd**(8), **pac**(8)

System and Network Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

proto – prototype job file for at

SYNOPSIS

/var/spool/cron/.proto

/var/spool/cron/.proto.*queue*

DESCRIPTION

When a job is submitted to **at** or **batch**, (see **at**(1)) the job is constructed as a shell script. First, a prologue is constructed, consisting of:

- A header specifying the owner, job name, and shell that should be used to run the job, and a flag indicating whether mail should be sent when the job completes;
- A set of Bourne shell commands to make the environment (see **environ**(5V)) for the **at** job the same as the current environment;
- A command to run the user's shell (as specified by the **SHELL** environment variable) with the rest of the job file as input.

at then reads a "prototype file," and constructs the rest of the job file from it.

Text from the prototype file is copied to the job file, except for special "variables" that are replaced by other text:

$d	is replaced by the current working directory
$l	is replaced by the current file size limit (see **ulimit**(3C))
$m	is replaced by the current umask (see **umask**(2V))
$t	is replaced by the time at which the job should be run, expressed as seconds since January 1, 1970, 00:00 Greenwich Mean Time, preceded by a colon
$<	is replaced by text read by **at** from the standard input (that is, the commands provided to **at** to be run in the job)

If the job is submitted in queue *queue*, **at** uses the file **/var/spool/cron/.proto.*queue*** as the prototype file if it exists, otherwise it will use the file **/var/spool/cron/.proto**.

EXAMPLES

The standard **.proto** file supplied with SunOS is:

```
#
# @(#)proto.5 1.3 89/10/05 SMI; from S5R3 1.1
#
cd $d
umask $m
$<
```

which causes commands to change the current directory in the job to the current directory at the time **at** was run, and to change the umask in the job to the umask at the time **at** was run, to be inserted before the commands in the job.

FILES

/var/spool/cron/.proto
/var/spool/cron/.proto.*queue*

SEE ALSO

at(1)

NAME

protocols – protocol name data base

SYNOPSIS

/etc/protocols

DESCRIPTION

The **protocols** file contains information regarding the known protocols used in the TCP/IP. For each protocol a single line should be present with the following information:

official-protocol-name protocol-number aliases

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Protocol names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

EXAMPLE

The following example is taken from SunOS.

```
#
# Internet (IP) protocols
#
ip      0     IP      # internet protocol, pseudo protocol number
icmp    1     ICMP    # internet control message protocol
ggp     3     GGP     # gateway-gateway protocol
tcp     6     TCP     # transmission control protocol
pup     12    PUP     # PARC universal packet protocol
udp     17    UDP     # user datagram protocol
```

FILES

/etc/protocols

SEE ALSO

getprotoent(3N)

BUGS

A name server should be used instead of a static file. A binary indexed file format should be available for fast access.

NAME

publickey – public key database

SYNOPSIS

/etc/publickey

DESCRIPTION

/etc/publickey is the public key database used for secure networking. Each entry in the database consists of a network user name (which may either refer to a user or a hostname), followed by the user's public key (in hex notation), a colon, and then the user's secret key encrypted with its login password (also in hex notation).

This file is altered either by the user through the **chkey**(1) command or by the system administrator through the **newkey**(8) command. The file **/etc/publickey** should only contain data on the Network Information Service (NIS) master machine, where it is converted into the NIS database **publickey.byname**.

The **/etc/publickey** file contains a default entry for **nobody**. If this entry is commented out, **chkey** only allows user to edit their existing entry, it will not allow them to create new entries.

SEE ALSO

chkey(1), **publickey**(3R), **newkey**(8), **ypupdated**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

queuedefs – queue description file for at, batch, and cron

SYNOPSIS

/var/spool/cron/queuedefs

DESCRIPTION

The **queuedefs** file describes the characteristics of the queues managed by **cron**(8). Each non-comment line in this file describes one queue. The format of the lines are as follows:

q.[*njob***j**][*nice***n**][*nwait***w**]

The fields in this line are:

q The name of the queue. **a** is the default queue for jobs started by **at**(1); **b** is the default queue for jobs started by **batch** (see **at**(1)); **c** is the default queue for jobs run from a **crontab**(5) file.

njob The maximum number of jobs that can be run simultaneously in that queue; if more than *njob* jobs are ready to run, only the first *njob* jobs will be run, and the others will be run as jobs that are currently running terminate. The default value is 100.

nice The **nice**(1) value to give to all jobs in that queue that are not run with a user ID of super-user. The default value is 2.

nwait The number of seconds to wait before rescheduling a job that was deferred because more than *njob* jobs were running in that job's queue, or because more than 25 jobs were running in all the queues. The default value is 60.

Lines beginning with # are comments, and are ignored.

EXAMPLE

```
#
# @(#)queuedefs 1.1 87/02/18 SMI; from S5R3
#
a.4j1n
b.2j2n90w
```

This file specifies that the **a** queue, for **at** jobs, can have up to 4 jobs running simultaneously; those jobs will be run with a **nice** value of 1. As no *nwait* value was given, if a job cannot be run because too many other jobs are running **cron** will wait 60 seconds before trying again to run it. The **b** queue, for **batch** jobs, can have up to 2 jobs running simultaneously; those jobs will be run with a **nice** value of 2. If a job cannot be run because too many other jobs are running, **cron** will wait 90 seconds before trying again to run it. All other queues can have up to 100 jobs running simultaneously; they will be run with a **nice** value of 2, and if a job cannot be run because too many other jobs are running **cron** will wait 60 seconds before trying again to run it.

FILES

/var/spool/cron/queuedefs

SEE ALSO

at(1), **nice**(1), **crontab**(5), **cron**(8)

NAME

rasterfile – Sun's file format for raster images

SYNOPSIS

#include <rasterfile.h>

DESCRIPTION

A rasterfile is composed of three parts: first, a header containing 8 integers; second, a (possibly empty) set of colormap values; and third, the pixel image, stored a line at a time, in increasing *y* order. The image is layed out in the file as in a memory pixrect. Each line of the image is rounded up to the nearest 16 bits.

The header is defined by the following structure:

```
struct rasterfile {
        int     ras_magic;
        int     ras_width;
        int     ras_height;
        int     ras_depth;
        int     ras_length;
        int     ras_type;
        int     ras_maptype;
        int     ras_maplength;
};
```

The *ras_magic* field always contains the following constant:

```
#define RAS_MAGIC     0x59a66a95
```

The *ras_width*, *ras_height*, and *ras_depth* fields contain the image's width and height in pixels, and its depth in bits per pixel, respectively. The depth is either 1 or 8, corresponding to standard frame buffer depths. The *ras_length* field contains the length in bytes of the image data. For an unencoded image, this number is computable from the *ras_width*, *ras_height*, and *ras_depth* fields, but for an encoded image it must be explicitly stored in order to be available without decoding the image itself. Note: the length of the header and of the (possibly empty) colormap values are not included in the value of the *ras_length* field; it is only the image data length. For historical reasons, files of type RT_OLD will usually have a 0 in the *ras_length* field, and software expecting to encounter such files should be prepared to compute the actual image data length if needed. The *ras_maptype* and *ras_maplength* fields contain the type and length in bytes of the colormap values, respectively. If *ras_maptype* is not RMT_NONE and the *ras_maplength* is not 0, then the colormap values are the *ras_maplength* bytes immediately after the header. These values are either uninterpreted bytes (usually with the *ras_maptype* set to RMT_RAW) or the equal length red, green and blue vectors, in that order (when the *ras_maptype* is RMT_EQUAL_RGB). In the latter case, the *ras_maplength* must be three times the size in bytes of any one of the vectors.

SEE ALSO

SunView Programmer's Guide

NAME

remote – remote host description file

SYNOPSIS

/etc/remote

DESCRIPTION

The systems known by **tip**(1C) and their attributes are stored in an ASCII file which is structured somewhat like the **termcap**(5) file. Each line in the file provides a description for a single *system*. Fields are separated by a colon ':'. Lines ending in a '\' character with an immediately following NEWLINE are continued on the next line.

The first entry is the name(s) of the host system. If there is more than one name for a system, the names are separated by vertical bars. After the name of the system comes the fields of the description. A field name followed by an '=' sign indicates a string value follows. A field name followed by a '#' sign indicates a following numeric value.

Entries named **tip***baudrate* are used as default entries by **tip**, as follows. When **tip** is invoked with only a phone number, it looks for an entry of the form **tip***baudrate*, where *baudrate* is the baud rate with which the connection is to be made. For example, if the connection is to be made at 300 baud, **tip** looks for an entry of the form **tip300**.

CAPABILITIES

Capabilities are either strings **(str)**, numbers **(num)**, or boolean flags **(bool)**. A string capability is specified by *capability=value*; for example, **'dv=/dev/harris'**. A numeric capability is specified by *capability#value*; for example, **'xa#99'**. A boolean capability is specified by simply listing the capability.

at **(str)** Auto call unit type. The following lists valid **'at'** types and their corresponding hardware:

biz31f	Bizcomp 1031, tone dialing
biz31w	Bizcomp 1031, pulse dialing
biz22f	Bizcomp 1022, tone dialing
biz22w	Bizcomp 1022, pulse dialing
df02	DEC DF02
df03	DEC DF03
ventel	Ventel 212+
v3451	Vadic 3451 Modem
v831	Vadic 831
hayes	Any Hayes-compatible modem
at	Any Hayes-compatible modem

br **(num)** The baud rate used in establishing a connection to the remote host. This is a decimal number. The default baud rate is 300 baud.

cm **(str)** An initial connection message to be sent to the remote host. For example, if a host is reached through a port selector, this might be set to the appropriate sequence required to switch to the host.

cu **(str)** Call unit if making a phone call. Default is the same as the **dv** field.

di **(str)** Disconnect message sent to the host when a disconnect is requested by the user.

du **(bool)** This host is on a dial-up line.

dv **(str)** Device(s) to open to establish a connection. If this file refers to a terminal line, **tip** attempts to perform an exclusive open on the device to insure only one user at a time has access to the port.

ec **(bool)** Initialize the **tip** variable **echocheck** to *on*, so that **tip** will synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted.

el **(str)** Characters marking an end-of-line. The default is no characters. **tip** only recognizes '~' escapes after one of the characters in **el**, or after a RETURN.

es **(str)** The command prefix (escape) character for **tip**.

et **(num)** Number of seconds to wait for an echo response when echo-check mode is on. This is a decimal number. The default value is 10 seconds.

ex **(str)** Set of non-printable characters not to be discarded when scripting with beautification turned on. The default value is "\t\n\b\f".

fo **(str)** Character used to force literal data transmission. The default value is '\377'.

fs **(num)** Frame size for transfers. The default frame size is equal to 1024.

hd **(bool)** Initialize the **tip** variable **halfduplex** to *on*, so local echo should be performed.

hf **(bool)** Initialize the **tip** variable **hardwareflow** to *on*, so hardware flow control is used.

ie **(str)** Input end-of-file marks. The default is a null string ("").

nb **(bool)** Initialize the **tip** variable **beautify** to *off*, so that unprintable characters will not be discarded when scripting.

nt **(bool)** Initialize the **tip** variable **tandem** to *off*, so that XON/XOFF flow control will not be used to throttle data from the remote host.

nv **(bool)** Initialize the **tip** variable **verbose** to *off*, so that verbose mode will be turned on.

oe **(str)** Output end-of-file string. The default is a null string (""). When **tip** is transferring a file, this string is sent at end-of-file.

pa **(str)** The type of parity to use when sending data to the host. This may be one of **even**, **odd**, **none**, **zero** (always set bit 8 to zero), **one** (always set bit 8 to 1). The default is **none**.

pn **(str)** Telephone number(s) for this host. If the telephone number field contains an '**@**' sign, **tip** searches the **/etc/phones** file for a list of telephone numbers — see **phones**(5). A '**%**' sign in the telephone number indicates a 5-second delay for the Ventel Modem.

pr **(str)** Character that indicates end-of-line on the remote host. The default value is '\n'.

ra **(bool)** Initialize the **tip** variable **raise** to *on*, so that lower case letters are mapped to upper case before sending them to the remote host.

rc **(str)** Character that toggles case-mapping mode. The default value is '\377'.

re **(str)** The file in which to record session scripts. The default value is **tip.record**.

rw **(bool)** Initialize the **tip** variable **rawftp** to *on*, so that all characters will be sent as is during file transfers.

sc **(bool)** Initialize the **tip** variable **script** to *on*, so that everything transmitted by the remote host will be recorded.

tb **(bool)** Initialize the **tip** variable **tabexpand** to *on*, so that tabs will be expanded to spaces during file transfers.

tc **(str)** Indicates that the list of capabilities is continued in the named description. This is used primarily to share common capability information.

Here is a short example showing the use of the capability continuation feature:

```
UNIX-1200:\
        :dv=/dev/cua0:el=^D^U^C^S^Q^O@:du:at=ventel:ie=#$%:oe=^D:br#1200:
arpavax|ax:\
        :pn=7654321%:tc=UNIX-1200
```

FILES

/etc/remote
/etc/phones

SEE ALSO

tip(1C), **phones**(5), **termcap**(5)

NAME

resolv.conf – configuration file for domain name system resolver

DESCRIPTION

The resolver configuration file contains information that is read by the domain name system resolver library the first time it is invoked in a process. It is only necessary to create this file to specify an explicit default domain name other than the default one derived from the **domainname**(1) command, or to specify name servers to use on other machines. The file is designed to be human readable and contains a list of keyword-value pairs that provide various types of resolver information.

keyword value

The different configuration options are:

nameserver *address* The Internet address (in dot notation) of a name server that the resolver should query. Up to MAXNS (currently 3) name servers may be listed. In that case the resolver library queries tries them in the order listed. The policy used is to try a name server, and if the query times out, try the next, until out of name servers, then repeat trying all the name servers until a maximum number of retries are made. If there are no **nameserver** lines in this file, then the loopback address is used, so there must be a name server running on the same machine.

domain *name* The default domain to append to names that do not have a dot in them, and used in searches. If there is no **domain** line in this file, then it is derived from the domain set by the **domainname**(1) command, usually by removing the first component. For example, if the **domainname**(1) is set to "foo.podunk.edu" then the default domain used by the resolver will be "podunk.edu". The is the same policy used by **sendmail**(8).

The keyword-value pair must appear on a single line, and the keyword (for instance, **nameserver**) must start the line. The value follows the keyword, separated by white space.

FILES

/etc/resolv.conf

SEE ALSO

domainname(1), **gethostent**(3N), **resolver**(3), **named**(8C), **nslookup**(8C), RFC 1034, RFC 1035

System and Network Administration

NAME

rfmaster – Remote File Sharing name server master file

SYNOPSIS

/usr/nserve/rfmaster

DESCRIPTION

The **rfmaster** file is an ASCII text file that identifies the hosts that are responsible for providing primary and secondary domain name service for Remote File Sharing domains. This file contains a series of entries, each terminated by a NEWLINE; a record may be extended over more than one line by escaping the NEWLINE with a backslash. Fields in each record are separated by white space. Each record has three fields: *name*, *type*, and *data*.

The *type* field, which defines the meaning of the *name* and *data* fields, has three possible values:

p Primary domain name server. In this case, *name* is the domain name and *data* is the full hostname of the primary name server, specified as:

domain.nodename

There can be only one primary name server per domain.

s Define a secondary name server for a domain. In this case, *name* and *data* are the same as for the **p** type. The order of the **s** entries in the **rfmaster** file determines the order in which secondary name servers take over when the current domain name server fails.

a Define a network address for a machine. In this case, *name* is the full domain name for the machine, and *data* is the network address. The network address can be in plain ASCII text or it can be preceded by a '\x' to be interpreted as hexadecimal notation.

There are at least two lines in the **rfmaster** file per domain name server: one **p** line and one **a** line. Together, they define the primary and its network address. There should also be at least one secondary name server in each domain.

This file is created and maintained on the primary domain name server. When a machine other than the primary tries to start Remote File Sharing, this file is read to determine the address of the primary. If this file is missing, the **–p** option of **rfstart** must be used to identify the primary. After that, a copy of the primary's **rfmaster** file is automatically placed on the machine.

Domains not served by the primary can also be listed in the **rfmaster** file. By adding primary, secondary, and address information for other domains on a network, machines served by the primary will be able to share resources with machines in other domains.

A primary name server may be a primary for more than one domain. However, the secondaries must then also be the same for each domain served by the primary.

EXAMPLE

An example of an **rfmaster** file is shown below. The network addresses given in the example are IP addresses; for more information on their format and how to generate them, see **hostrfs**(8).

```
sunrfs          p    sunrfs.estale
sunrfs          s    sunrfs.ivy
sunrfs.estale   a    \x000214508190320d
sunrfs.ivy      a    \x0002145081903246
```

Note: If a line in the **rfmaster** file begins with a '#' (pound sign) character, the entire line will be treated as a comment.

FILES

/usr/nserve/rfmaster

SEE ALSO

rfstart(8)

System and Network Administration

NAME

rgb – available colors (by name) for coloredit

SYNOPSIS

.rgb

$HOME/.rgb

/usr/lib/.rgb

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

.rgb is an ASCII file containing consecutive lines terminated by newlines. Each line starts with three integers, each in the range 0-255. These integers are the RGB equivalent for the color named on the same line. At least one tab character delimits the last integer from the name field. The coloreditor searches for this file, first in the current directory; next, in the users home directory; and finally, in **/usr/lib**. The user can add to or delete from the **.rgb** file that he or she has access to, thus changing the available color table for subsequent invocations of coloredit.

EXAMPLES

The following is an example of a **.rgb** file.

0 0 0	**Black**
0 0 255	**Blue**
95 159 159	**Cadet Blue**
66 66 111	**Cornflower Blue**
107 35 142	**Dark Slate Blue**

SEE ALSO

coloredit(1)

NAME

rootmenu – root menu specification for SunView

SYNOPSIS

~/.rootmenu
/usr/lib/.rootmenu

DESCRIPTION

If a **.rootmenu** file is present in a user's home directory, it specifies the SunView menu, the menu that appears when the user clicks and holds the right mouse button in the background of the SunView desktop. If a **.rootmenu** file is not present in the user's home directory, **/usr/lib/.rootmenu** specifies the SunView menu.

Each line of a **.rootmenu** file has the format:

menu item *command*

menu item can be a character string, or an icon file delimited by angle brackets:

<icon-filename>

If *menu item* is a character string with embedded spaces, it must be enclosed by double quotes ('"').

command can be a command line to be executed when the menu item is selected, or one of the following reserved-word commands:

EXIT	Exit **sunview** (requires confirmation).
REFRESH	Redraw the entire screen.
MENU	This menu item is a pull-right item with a submenu. If a full pathname follows the **MENU** command, the submenu contents are taken from that file. Otherwise, all the lines between a **MENU** command and a matching **END** command are added to the submenu.
END	Mark the end of a nested submenu. The left side of this line should match the left side of a line with a **MENU** command.

If *command* is not one of the reserved-word commands, it is treated as a command line, although no shell interpretation is done.

Lines beginning with a '#' character are considered comments and are ignored.

If a user's **.rootmenu** file is modified, the SunView menu immediately reflects the changes.

See **sunview**(1) for more details about **.rootmenu**.

EXAMPLES

The following is a sample **.rootmenu** file:

```
#
#       sample root menu
#
"Lock Screen"           lockscreen
Tools   MENU
        Perfmeter               perfmeter
        Calculator              calc
        Mailtool                mailtool
Tools   END
"ShellTool"             shelltool
"CommandTool"           cmdtool
"Console"               cmdtool -C
#"MailTool"             mailtool
"TextEditor"            textedit
```

```
        "DefaultsEditor"        defaultsedit
        #"IconEditor"           iconedit
        #"DbxTool"              dbxtool
        "PerfMeter"             perfmeter
        #"GraphicsTool"         gfxtool
        "Redisplay All"         REFRESH
        "Exit Suntools"         EXIT
```

SEE ALSO

sunview(1)

NAME

rpc – rpc program number data base

SYNOPSIS

/etc/rpc

DESCRIPTION

The **rpc** file contains user readable names that can be used in place of rpc program numbers. Each line has the following information:

rpc-program-server *rpc-program-number* *aliases*

Items are separated by any number of blanks and/or tab characters. A "#" indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

EXAMPLE

Here is an example of the **/etc/rpc** file from the SunOS System.

```
#
#       rpc  1.10  87/04/10
#
portmapper      100000    portmap sunrpc
rstatd          100001    rstat rup perfmeter
rusersd         100002    rusers
nfs             100003    nfsprog
ypserv          100004    ypprog
mountd          100005    mount showmount
ypbind          100007
walld           100008    rwall shutdown
yppasswdd       100009    yppasswd
etherstatd      100010    etherstat
rquotad         100011    rquotaprog quota rquota
sprayd          100012    spray
3270_mapper     100013
rje_mapper      100014
selection_svc   100015    selnsvc
database_svc    100016
rexd            100017    rex
alis            100018
sched           100019
llockmgr        100020
nlockmgr        100021
x25.inr         100022
statmon         100023
status          100024
bootparam       100026
ypupdated       100028    ypupdate
keyserv         100029    keyserver
```

FILES

/etc/rpc

SEE ALSO

getrpcent(3N)

NAME

sccsfile – format of an SCCS history file

DESCRIPTION

An SCCS file is an ASCII file consisting of six logical parts:

checksum	character count used for error detection
delta table	log containing version info and statistics about each delta
usernames	login names and/or group IDs of users who may add deltas
flags	definitions of internal keywords
comments	arbitrary descriptive information about the file
body	the actual text lines intermixed with control lines

Each section is described in detail below.

Conventions

Throughout an SCCS file there are lines which begin with the ASCII SOH (start of heading) character (octal 001). This character is hereafter referred to as the *control character*, and will be represented as '^A'. If a line described below is not depicted as beginning with the control character, it cannot do so and still be within SCCS file format.

Entries of the form *ddddd* represent a five digit string (a number between 00000 and 99999).

Checksum

The checksum is the first line of an SCCS file. The form of the line is:

^A **h***ddddd*

The value of the checksum is the sum of all characters, except those contained in the first line. The ^A**h** provides a *magic number* of (octal) 064001.

Delta Table

The delta table consists of a variable number of entries of the form:

^A**s** *inserted/deleted/unchanged*
^A**d** *type sid yr/mo/da hr:mi:se username serial-number predecessor-sn*
^A**i** *include-list*
^A**x** *exclude-list*
^A**g** *ignored-list*
^A**m** *mr-number*
...
^A**c** *comments ...*
...
^A**e**

The first line (^A**s**) contains the number of lines inserted/deleted/unchanged respectively. The second line (^A**d**) contains the type of the delta (normal: **D**, and removed: **R**), the SCCS ID of the delta, the date and time of creation of the delta, the user-name corresponding to the real user ID at the time the delta was created, and the serial numbers of the delta and its predecessor, respectively.

The ^A**i**, ^A**x**, and ^A**g** lines contain the serial numbers of deltas included, excluded, and ignored, respectively. These lines do not always appear.

The ^A**m** lines (optional) each contain one MR number associated with the delta; the ^A**c** lines contain comments associated with the delta.

The ^A**e** line ends the delta table entry.

User Names

The list of user-names and/or numerical group IDs of users who may add deltas to the file, separated by NEWLINE characters. The lines containing these login names and/or numerical group IDs are surrounded by the bracketing lines ^**Au** and ^**AU**. An empty list allows anyone to make a delta.

Flags

Flags are keywords that are used internally (see **sccs-admin**(1) for more information on their use). Each flag line takes the form:

^**Af** *flag optional text*

The following flags are defined in order of appearance:

^**Af** t *type-of-program*
Defines the replacement for the **%T%** ID keyword.

^**Af** v program-name
Controls prompting for MR numbers in addition to comments; if the optional text is present it defines an MR number validity checking program.

^**Af** i Indicates that the '**No id keywords**' message is to generate an error that terminates the SCCS command. Otherwise, the message is treated as a warning only.

^**Af** b Indicates that the **–b** option may be used with the SCCS **get** command to create a branch in the delta tree.

^**Af** m module name
Defines the first choice for the replacement text of the **%M%** ID keyword.

^**Af** f *floor*
Defines the "floor" release; the release below which no deltas may be added.

^**Af** c *ceiling*
Defines the "ceiling" release; the release above which no deltas may be added.

^**Af** d *default-sid*
The **d** flag defines the default SID to be used when none is specified on an SCCS **get** command.

^**Af** n The **n** flag enables the SCCS **delta** command to insert a "null" delta (a delta that applies *no* changes) in those releases that are skipped when a delta is made in a *new* release (for example, when delta 5.1 is made after delta 2.7, releases 3 and 4 are skipped).

^**Af** j Enables the SCCS **get** command to allow concurrent edits of the same base SID.

^**Af** l *lock-releases*
Defines a *list* of releases that are locked against editing.

^**Af** q user defined
Defines the replacement for the **%Q%** ID keyword.

^**Af** e **0**|**1**
The **e** flag indicates whether a source file is encoded or not. A **1** indicates that the file is encoded. Source files need to be encoded when they contain control characters, or when they do not end with a NEWLINE. The **e** flag allows files that contain binary data to be checked in.

Comments

Arbitrary text surrounded by the bracketing lines ^**At** and ^**AT**. The comments section typically will contain a description of the file's purpose.

Body

The body consists of text lines and control lines. Text lines do not begin with the control character, control lines do. There are three kinds of control lines: *insert*, *delete*, and *end*, represented by:

^**AI** *ddddd*
^**AD** *ddddd*
^**AE** *ddddd*

respectively. The digit string is the serial number corresponding to the delta for the control line.

SEE ALSO

sccs(1), **sccs-admin**(1), **sccs-cdc**(1), **sccs-comb**(1), **sccs-delta**(1), **sccs-get**(1), **sccs-help**(1), **sccs-prs**(1), **sccs-prt**(1), **sccs-rmdel**(1), **sccs-sact**(1), **sccs-sccsdiff**(1), **sccs-unget**(1), **sccs-val**(1), **what**(1)

NAME

services – Internet services and aliases

DESCRIPTION

The **services** file contains an entry for each service available through the TCP/IP. Each entry consists of a line of the form:

service-name port/protocol aliases

service-name This is the official Internet service name.

port/protocol This field is composed of the port number and protocol through which the service is provided (for instance, **512/tcp**).

aliases This is a list of alternate names by which the service might be requested.

Fields can be separated by any number of spaces or TAB's. A '#' (pound-sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Service names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

FILES

/etc/services

SEE ALSO

getservent(3N), **inetd.conf**(5)

BUGS

A name server should be used instead of a static file.

NAME

setup.pc – master configuration file for DOS

SYNOPSIS

~/pc/setup.pc

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **setup.pc** file in your home PC directory, **~/pc**, is the master configuration file for DOS. Changes to the file take effect for all new DOS windows you start. The definitions made in **setup.pc** and **AUTOEXEC.BAT** serve to define your system to DOS. Among other things, the **setup.pc** file defines:

- The printers or devices to which you assign DOS printer names (LPT1, LPT2, LPT3)
- The devices or boards that are tied to the DOS communications devices (COM1, COM2)
- The name of a special DOS quick-start file that you may have set up
- The drive C file to be used

The format of each line is as follows; separators can be TAB or SPACE characters:

DOS Device *SunOS Device or Command*

DOS Device

The name of the device as DOS knows it. For example, the device name for the first diskette drive in DOS is "A".

SunOS Device or Command

The name of the device as the SunOS system knows it. This can also be a symbolic link to the real device name. For example, **/etc/dos/defaults/diskette_a** is a symbolic link to **/dev/rfd0c**. For emulated DOS printers (LPT1, LPT2, or LPT3), specify a command or command pipeline.

EXAMPLES

```
# DOS Device    SunOS Device Path Name
#
A               /etc/dos/defaults/diskette_a
#B              /etc/dos/defaults/diskette_b
C               ~/pc/C:
COM1            /etc/dos/defaults/com1
#COM2           /etc/dos/defaults/com2
LPT1            lpr
LPT2            cat >>~/lpt-2
LPT3            psfx80 | lpr
SAVE            ~/pc/.quickpc
#CMDTOOL
#TEXT
#BOARDS
```

Placed at the beginning of a line to indicate a comment.

A, B Diskette drivers defined using the standard SunOS names for the Sun386i diskette drives. Drive **A** is normally assigned to the built-in diskette drive.

C The emulated C drive. It is actually stored as one large system file.

COM1, COM2
Serial ports. The first DOS serial port (COM1) is assigned to the Sun386i built-in serial port. To use the built-in serial port as COM2, comment out the COM1 line and uncomment the COM2 line. (DOS Windows directs the output of either COM1 or COM2 to the built-in port, but uses different interrupt levels so that COM2 "appears" to DOS to be a second serial port.) You can also add a real second serial port by installing an AT or XT card and enabling the SunOS ATS driver.

LPT1, LPT2, LPT3
Emulated printers. DOS printer names can be assigned to SunOS printers, other devices, or files.

SAVE The "quick-start" file DOS reads at startup for faster loading.

CMDTOOL
Used to list the SunOS commands that must run in a separate Commands window when started from DOS. The following SunOS commands automatically run in a Commands window when you run them from DOS:

mail man more passwd rlogin stty vi

If there are other SunOS commands or applications you want to run from DOS, and these commands require keyboard entry or Commands window display, list them here. If you add entries to this line, separate them with a SPACE character, and be sure to remove the # (comment) symbol to activate the line.

TEXT Specifies a list of "text-only" DOS programs. Such programs do not require a PC window because they do not print at specific screen positions; they can print text in a current Commands window if that is where you are working at the time. An example is a C compiler or a linker that runs from the DOS command line. If you place entries on this line, separate them with a SPACE character, and be sure to remove the # symbol to activate the text-only line.

BOARDS
A list of boards that DOS should attempt to activate when opening a DOS window. Each board you list here must have a corresponding entry in the **boards.pc file** (see **boards.pc**(5)).

You can create task-specific DOS environments by setting up additional **setup.pc** files to attach different printers, drive C files, and other real and emulated devices.

If you are installing a board that duplicates a function normally enabled in the **setup.pc** file, you should disable the corresponding **setup.pc** line by commenting it out with #.

FILES

~/pc/setup.pc	Personal **setup.pc** file, copied to the user's **pc** directory when DOS is started for the first time.
/etc/dos/defaults/setup.pc	Master copy of **setup.pc** for the workstation.

SEE ALSO
dos(1), **boards.pc**(5)

Sun386i User's Guide,
Sun386i Advanced Skills,
Sun MS-DOS Reference Manual

NAME

sm, sm.bak, sm.state – in.statd directory and file structures

SYNOPSIS

/etc/sm, /etc/sm.bak, /etc/sm.state

DESCRIPTION

/etc/sm and **/etc/sm.bak** are directories generated by **in.statd**. Each entry in **/etc/sm** represents the name of the machine to be monitored by the **in.statd** daemon. Each entry in **/etc/sm.bak** represents the name of the machine to be notified by the **in.statd** daemon upon its recovery.

/etc/sm.state is a file generated by **rpc.statd** to record the its version number. This version number is incremented each time a crash or recovery takes place.

FILES

/etc/sm
/etc/sm.bak
/etc/sm.state

SEE ALSO

lockd(8C), **statd**(8C)

NAME

sm, sm.bak, state – statd directories and file structures

SYNOPSIS

/etc/sm /etc/sm.bak /etc/state

DESCRIPTION

/etc/sm and **/etc/sm.bak** are directories generated by **statd**. Each entry in **/etc/sm** represents the name of the machine to be monitored by the **statd** daemon. Each entry in **/etc/sm.bak** represents the name of the machine to be notified by the **statd** daemon upon its recovery.

/etc/state is a file generated by **statd** to record its version number. This version number is incremented each time a crash or recovery takes place.

FILES

/etc/sm
/etc/sm.bak
/etc/state

SEE ALSO

lockd(8C), **statd**(8C)

NAME

sunview – initialization file for SunView

SYNOPSIS

~/.sunview
~/.suntools
/usr/lib/.sunview

DESCRIPTION

If the file **.sunview** or **.suntools** is present in a user's home directory when the user starts up **sunview**(1), **sunview** starts up the "tools", or window-based applications listed in this file. You can use a **.sunview** or **.suntools** file to customize your desktop layout. If a **.sunview** or **.suntools** file is not present in the user's home directory, **sunview** starts up the tools listed in **/usr/lib/.sunview.**

Each line of a **.sunview** or **.suntools** file has the format:

SunView-tool [*options*]

SunView-tool is in the form of a command line, although no shell interpretation is done. *options* are command line options which may include SunView generic tool arguments (see **sunview**(1) for a description of generic tool arguments). Lines beginning with the '#' character are considered comments and are ignored.

EXAMPLES

Here is a sample **.sunview** file:

```
#
#       sample .sunview file
#
cmdtool  -Wp  0  0 -WP   0  0 -Wh  3 -Ww 80 -Wl "<< CONSOLE >>" -WL "console" -C
clock    -Wp 497  32 -WP  704  0 -Wi -Wh 1
cmdtool  -Wp  0  71 -WP  772  0 -Wi -Wh 44 -Ww 80
textedit -Wp 259  98 -WP  840  0 -Wi
mailtool -Wp 492  71 -WP  908  0 -Wi
```

SEE ALSO

sunview(1), **toolplaces**(1)

NAME

svdtab – SunView device table

SYNOPSIS

/etc/svdtab

DESCRIPTION

The **/etc/svdtab** contains information that is used by the window system (for example, **sunview**(1)) to change the owner, group, and permissions of the window devices (**/dev/win***) upon startup. By default *all* lines in this file are commented out. That is, all security is disabled. To enable security, uncomment the following line in **/etc/svdtab** and start up the window system again:

#0600

If **/etc/svdtab** contains an entry, the owner and group of the **win** devices are set to the owner and group of the console. The permissions are set as specified in **/etc/svdtab**. The recommended permissions for normal security is **0600**.

Once the window devices are owned by the user, their permissions and ownership can be changed using **chmod**(1V) and **chown**(8), as with any user-other file.

EXAMPLES

The following is an example entry of the **/etc/svdtab** file:

0600

This entry specifies that upon SunView startup, the owner, group and permissions of **/dev/win*** will be set to the user's username, the user's group and **0600**, respectively. Upon exiting the window system, the owner and group of **/dev/win***, will be reset to **root**, and **wheel**. The permissions remain as set in **/etc/svdtab**. If no entry appears in this file, the owner, group and permissions will *not* be changed.

SEE ALSO

chmod(1V), **sunview**(1), **chown**(8)

NOTES

If the window system dies unnaturally, for example by **kill**(1), the owner, group and permissions remain as set when the window was started up.

NAME

syslog.conf – configuration file for syslogd system log daemon

SYNOPSIS

/etc/syslog.conf

DESCRIPTION

The file **/etc/syslog.conf** contains information used by the system log daemon, **syslogd**(8), to forward a system message to appropriate log files and/or users. **syslog** preprocesses this file through **m4**(1V) to obtain the correct information for certain log files.

A configuration entry is composed of two TAB-separated fields:

selector *action*

The *selector* field contains a semicolon-separated list of priority specifications of the form:

facility.level[;*facility.level*]

where *facility* is a system facility, or comma-separated list of facilities, and *level* is an indication of the severity of the condition being logged. Recognized values for *facility* include:

user Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file.

kern Messages generated by the kernel.

mail The mail system.

daemon System daemons, such as **ftpd**(8C), **routed**(8C), etc.

auth The authorization system: **login**(1), **su**(1V), **getty**(8), etc.

lpr The line printer spooling system: **lpr**(1), **lpc**(8), **lpd**(8), etc.

news Reserved for the USENET network news system.

uucp Reserved for the UUCP system; it does not currently use the **syslog** mechanism.

cron The **cron/at** facility; **crontab**(1), **at**(1), **cron**(8), etc.

local0-7 Reserved for local use.

mark For timestamp messages produced internally by **syslogd**.

***** An asterisk indicates all facilities except for the **mark** facility.

Recognized values for *level* are (in descending order of severity):

emerg For panic conditions that would normally be broadcast to all users.

alert For conditions that should be corrected immediately, such as a corrupted system database.

crit For warnings about critical conditions, such as hard device errors.

err For other errors.

warning For warning messages.

notice For conditions that are not error conditions, but may require special handling.

info Informational messages.

debug For messages that are normally used only when debugging a program.

none Do not send messages from the indicated *facility* to the selected file. For example, a *selector* of

***.debug;mail.none**

will send all messages *except* mail messages to the selected file.

The *action* field indicates where to forward the message. Values for this field can have one of four forms:

- A filename, beginning with a leading slash, which indicates that messages specified by the *selector* are to be written to the specified file. The file will be opened in append mode.
- The name of a remote host, prefixed with an **@**, as with: *@server*, which indicates that messages specified by the *selector* are to be forwarded to the **syslogd** on the named host.
- A comma-separated list of usernames, which indicates that messages specified by the *selector* are to be written to the named users if they are logged in.
- An asterisk, which indicates that messages specified by the *selector* are to be written to all logged-in users.

Blank lines are ignored. Lines for which the first character is a '#' are treated as comments.

Sun386i DESCRIPTION

The file is as described above, except that there is an additional valid entry type, for translation. A line containing the keyword "translate," if present, specifies how system error messages are translated, suppressed, or forwarded to appropriate log files and/or users.

A translation entry in the file is composed of five TAB-separated fields:

translate *source* *facility* *input* *output*

The *translate* field consists of the word **translate** and is used to indicate that this is a translation entry.

The *source* field contains a comma separated list of source names. Recognized sources are:

klog Messages placed in **/dev/klog** by the kernel.

log Messages placed in **/dev/log** file by local programs.

syslog Messages placed in the internet socket by programs on other systems.

***** An asterisk indicates all three sources (**klog**, **log** and **syslog**).

The *facility* field contains a comma-separated list of facilities.

The *input* field is the name of the file used to map error messages (in printf format strings) to numbers. This number is used to locate a new string in the file specified in the output field. The format of both files is described in **translate**(5).

The output file specified by the output field translates the numbers from the input file into the desired error messages, and also specifies the format to be used to output each message.

EXAMPLE

With the following configuration file:

***.notice;mail.info**	**/var/log/notice**
***.crit**	**/var/log/critical**
kern,mark.debug	**/dev/console**
kern.err	**@server**
***.emerg**	*****
***.alert**	**root,operator**
***.alert;auth.warning**	**/var/log/auth**

syslogd will log all mail system messages except **debug** messages and all **notice** (or higher) messages into a file named **/var/log/notice**. It logs all critical messages into **/var/log/critical**, and all kernel messages and 20-minute marks onto the system console.

Kernel messages of **err** (error) severity or higher are forwarded to the machine named *server*. Emergency messages are forwarded to all users. The users "root" and "operator" are informed of any **alert** messages. All messages from the authorization system of **warning** level or higher are logged in the file **/var/log/auth**.

FILES

/etc/syslog.conf
/var/log/notice
/var/log/critical
/var/log/auth

SEE ALSO

at(1), **crontab**(1), **logger**(1), **login**(1), **lpr**(1), **m4**(1V), **su**(1V), **syslog**(3), **translate**(5), **cron**(8), **ftpd**(8C), **getty**(8), **lpc**(8), **lpd**(8), **routed**(8C), **syslogd**(8)

NAME

systems – NIS systems file

SYNOPSIS

/etc/systems

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **/etc/systems** file is used only by SNAP and Automatic System Installation, and contains basic information about each host on the network. It is an ASCII file in the **/etc** directory on the Network Information Service (NIS) master server. To successfully administer all systems in a NIS domain using SNAP, there must be an entry in this file for each host listed in the **/etc/hosts** file. For each host, this file contains a one-line entry, of the following form, where each field *must* be separated by a TAB character:

system architecture sunos "hostid" memory_size disk_size network_role

system is the name of a host, whether it is on a network or a standalone system. This field contains only lowercase and numeric characters, must start with a lower-case character, and must not be longer than 32 characters.

architecture
indicates the architecture of the specified system. This can be **s386**, **sun4**, **sun3**, **sun2**, **sun1**, **pcnfs**, or **other**.

sunos indicates the SunOS operating system version the system is running. Typically, the form is **sunos***version_number* or **unknown**. SNAP always inserts **unknown** when adding new systems.

hostid the system host ID, as obtained from **/bin/hostid**. This entry must be in quotes. If the host ID is **unknown**, an empty string (" ") is specified. SNAP always inserts an empty string when adding new systems.

memory_size
amount of memory, in kilobytes. This can be **8000** (for 8 megabytes), **4000** (for 4 megabytes), or **–1** for **unknown**. SNAP always inserts **–1** when adding new systems.

disk_size
amount of disk space, in kilobytes. This can be any value, but typically should be close to the actual disk size or to the total amount of disk space, if expansion disks were added. Diskless clients would have a zero value, while **unknown** disk sizes are specified by a **–1** value. SNAP always inserts **–1** when adding new network clients.

network_role
indicates the role the system plays on the network. This can be **master_bootserver**, **slave_bootserver**, **network_client**, or **diskless_client**.

EXAMPLES

Here is a sample systems file:

```
vulcan     s386    sunos4.0.1    "12345678"    8000    327000    master_bootserver
polaris    s386    sunos4.0.1    ""            8000    91000     slave_bootserver
star       sun4    sunos4.1      ""            8000    91000     network_client
traveler   s386    sunos4.0.1    ""            8000    0         diskless_client
```

FILES

/etc/systems
/etc/hosts
/bin/hostid

SEE ALSO

snap(1), **vipw**(8)

System and Network Administration,
Sun386i Advanced Administration

NOTES

Take precautions to lock the **/etc/systems** file against simultaneous changes if it will be edited with a text editor; editing with **vipw**(8) provides the necessary locking.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

tar – tape archive file format

DESCRIPTION

tar, (the tape archive command) dumps several files into one, in a medium suitable for transportation.

A "tar tape" or file is a series of blocks. Each block is of size TBLOCK. A file on the tape is represented by a header block which describes the file, followed by zero or more blocks which give the contents of the file. At the end of the tape are two blocks filled with binary zeros, as an EOF indicator.

The blocks are grouped for physical I/O operations. Each group of *n* blocks (where *n* is set by the **b** keyletter on the **tar**(1) command line — default is 20 blocks) is written with a single system call; on nine-track tapes, the result of this write is a single tape record. The last group is always written at the full size, so blocks after the two zero blocks contain random data. On reading, the specified or default group size is used for the first read, but if that read returns less than a full tape block, the reduced block size is used for further reads, unless the **B** keyletter is used.

The header block looks like:

```
#define TBLOCK 512
#define NAMSIZ 100
union hblock {
        char dummy[TBLOCK];
        struct header {
                char name[NAMSIZ];
                char mode[8];
                char uid[8];
                char gid[8];
                char size[12];
                char mtime[12];
                char chksum[8];
                char linkflag;
                char linkname[NAMSIZ];
        } dbuf;
};
```

name is a null-terminated string. The other fields are zero-filled octal numbers in ASCII. Each field (of width *w*) contains w-2 digits, a SPACE, and a null character, except *size* and *mtime*, which do not contain the trailing null. *name* is the name of the file, as specified on the **tar** command line. Files dumped because they were in a directory which was named in the command line have the directory name as prefix and */filename* as suffix. *mode* is the file mode, with the top bit masked off. *uid* and *gid* are the user and group numbers which own the file. *size* is the size of the file in bytes. Links and symbolic links are dumped with this field specified as zero. *mtime* is the modification time of the file at the time it was dumped. *chksum* is a decimal ASCII value which represents the sum of all the bytes in the header block. When calculating the checksum, the *chksum* field is treated as if it were all blanks. *linkflag* is ASCII '0' if the file is "normal" or a special file, ASCII '1' if it is an hard link, and ASCII '2' if it is a symbolic link. The name linked-to, if any, is in *linkname*, with a trailing null character. Unused fields of the header are binary zeros (and are included in the checksum).

The first time a given inode number is dumped, it is dumped as a regular file. The second and subsequent times, it is dumped as a link instead. Upon retrieval, if a link entry is retrieved, but not the file it was linked to, an error message is printed and the tape must be manually re-scanned to retrieve the linked-to file.

The encoding of the header is designed to be portable across machines.

SEE ALSO

tar(1)

BUGS

Names or linknames longer than NAMSIZ produce error reports and cannot be dumped.

NAME

term – terminal driving tables for nroff

SYNOPSIS

/usr/lib/term/tab*name*

DESCRIPTION

nroff(1) uses driving tables to customize its output for various types of output devices, such as terminals, line printers, daisy-wheel printers, or special output filter programs. These driving tables are written as C programs, compiled, and installed in the directory **/usr/lib/term**. The *name* of the output device is specified with the **–T** option of **nroff**. The structure of the terminal table is as follows:

```
#define    INCH    240

struct {
        int bset;
        int breset;
        int Hor;
        int Vert;
        int Newline;
        int Char;
        int Em;
        int Halfline;
        int Adj;
        char *twinit;
        char *twrest;
        char *twnl;
        char *hlr;
        char *hlf;
        char *flr;
        char *bdon;
        char *bdoff;
        char *ploton;
        char *plotoff;
        char *up;
        char *down;
        char *right;
        char *left;
        char *codetab[256–32];
        char *zzz;
} t;
```

The meanings of the various fields are as follows:

bset Bits to set in the **sg_flags** field of the **sgtty** structure before output; see **ttcompat**(4M).

breset Bits to reset in the **sg_flags** field of the **sgtty** structure after output; see **ttcompat**(4M).

Hor Horizontal resolution in fractions of an inch.

Vert Vertical resolution in fractions of an inch.

Newline Space moved by a NEWLINE (LINEFEED) character in fractions of an inch.

Char Quantum of character sizes, in fractions of an inch (that is, a character is a multiple of **Char** units wide).

Em Size of an em in fractions of an inch.

Halfline Space moved by a half-LINEFEED (or half-reverse-LINEFEED) character in fractions of an inch.

Adj Quantum of white space, in fractions of an inch. (that is, white spaces are a multiple of **Adj** units wide)

Note: if this is less than the size of the SPACE character (in units of **Char**; see below for how the sizes of characters are defined), **nroff** will output fractional SPACE characters using plot mode. Also, if the **–e** switch to **nroff** is used, **Adj** is set equal to **Hor** by **nroff**.

twinit Set of characters used to initialize the terminal in a mode suitable for **nroff**.

twrest Set of characters used to restore the terminal to normal mode.

twnl Set of characters used to move down one line.

hlr Set of characters used to move up one-half line.

hlf Set of characters used to move down one-half line.

flr Set of characters used to move up one line.

bdon Set of characters used to turn on hardware boldface mode, if any.

bdoff Set of characters used to turn off hardware boldface mode, if any.

ploton Set of characters used to turn on hardware plot mode (for Diablo type mechanisms), if any.

plotoff Set of characters used to turn off hardware plot mode (for Diablo type mechanisms), if any.

up Set of characters used to move up one resolution unit (**Vert**) in plot mode, if any.

down Set of characters used to move down one resolution unit (**Vert**) in plot mode, if any.

right Set of characters used to move right one resolution unit (**Hor**) in plot mode, if any.

left Set of characters used to move left one resolution unit (**Hor**) in plot mode, if any.

codetab Definition of characters needed to print an **nroff** character on the terminal. The first byte is the number of character units (**Char**) needed to hold the character; that is, **\001** is one unit wide, **\002** is two units wide, etc. The high-order bit (0200) is on if the character is to be underlined in underline mode (**.ul**). The rest of the bytes are the characters used to produce the character in question. If the character has the sign (0200) bit on, it is a code to move the terminal in plot mode. It is encoded as:

0100 bit on	vertical motion.
0100 bit off	horizontal motion.
040 bit on	negative (up or left) motion.
040 bit off	positive (down or right) motion.
037 bits	number of such motions to make.

zzz A zero terminator at the end.

All quantities which are in units of fractions of an inch should be expressed as '**INCH***num*/*denom*', where *num* and *denom* are respectively the numerator and denominator of the fraction; that is, 1/48 of an inch would be written as '**INCH/48**'.

If any sequence of characters does not pertain to the output device, that sequence should be given as a null string.

The following is a sample **codetab** encoding.

```
"\001 ",                    /*space*/
"\001!",                    /*!*/
"\001\"",                   /*"*/
"\001#",                    /*#*/
"\001$",                    /*$*/
```

```
"\001%",        /*%*/
"\001&",        /*&*/
"\001'",        /*'*/
"\001(",        /*(*/
"\001)",        /*)*/
"\001*",        /***/
"\001+",        /*+*/
"\001,",        /*,*/
"\001-",        /*-*/
"\001.",        /*.*/
"\001/",        /*/*/
"\2010",        /*0*/
"\2011",        /*1*/
"\2012",        /*2*/
"\2013",        /*3*/
"\2014",        /*4*/
"\2015",        /*5*/
"\2016",        /*6*/
"\2017",        /*7*/
"\2018",        /*8*/
"\2019",        /*9*/
"\001:",        /*:*/
"\001;",        /*;*/
"\001<",        /*<*/
"\001=",        /*=*/
"\001>",        /*>*/
"\001?",        /*?*/
"\001@",        /*@*/
"\201A",        /*A*/
"\201B",        /*B*/
"\201C",        /*C*/
"\201D",        /*D*/
"\201E",        /*E*/
"\201F",        /*F*/
"\201G",        /*G*/
"\201H",        /*H*/
"\201I",        /*I*/
"\201J",        /*J*/
"\201K",        /*K*/
"\201L",        /*L*/
"\201M",        /*M*/
"\201N",        /*N*/
"\201O",        /*O*/
"\201P",        /*P*/
"\201Q",        /*Q*/
"\201R",        /*R*/
"\201S",        /*S*/
"\201T",        /*T*/
"\201U",        /*U*/
"\201V",        /*V*/
"\201W",        /*W*/
"\201X",        /*X*/
"\201Y",        /*Y*/
```

```
"\201Z",                    /*Z*/
"\001[",                    /*[*/
"\001\\",                   /*\*/
"\001]",                    /*]*/
"\001^",                    /*^*/
"\001_",                    /*_*/
"\001`",                    /*`*/
"\201a",                    /*a*/
"\201b",                    /*b*/
"\201c",                    /*c*/
"\201d",                    /*d*/
"\201e",                    /*e*/
"\201f",                    /*f*/
"\201g",                    /*g*/
"\201h",                    /*h*/
"\201i",                    /*i*/
"\201j",                    /*j*/
"\201k",                    /*k*/
"\201l",                    /*l*/
"\201m",                    /*m*/
"\201n",                    /*n*/
"\201o",                    /*o*/
"\201p",                    /*p*/
"\201q",                    /*q*/
"\201r",                    /*r*/
"\201s",                    /*s*/
"\201t",                    /*t*/
"\201u",                    /*u*/
"\201v",                    /*v*/
"\201w",                    /*w*/
"\201x",                    /*x*/
"\201y",                    /*y*/
"\201z",                    /*z*/
"\001{",                    /*{*/
"\001|",                    /*|*/
"\001}",                    /*}*/
"\001~",                    /*~*/
"\000\0",                   /*narrow sp*/
"\001-",                    /*hyphen*/
"\001\016Z\017",            /*bullet*/
"\002[]",                   /*square*/
"\002--",                   /*3/4 em dash*/
"\001_",                    /*rule*/
"\0031/4",                  /*1/4*/
"\0031/2",                  /*1/2*/
"\0033/4",                  /*3/4*/
"\001-",                    /*minus*/
"\202fi",                   /*fi*/
"\202fl",                   /*fl*/
"\202ff",                   /*ff*/
"\203ffi",                  /*ffi*/
"\203ffl",                  /*ffl*/
"\001\016p\017",            /*degree*/
```

```
"\001|\b\342-\302",          /*dagger*/
"\001\301s\343s\302",        /*section*/
"\001'",                     /*foot mark*/
"\001\033Z",                 /*acute accent*/
"\001`",                     /*grave accent*/
"\001_",                     /*underrule*/
"\001/",                     /*long slash*/
"\000\0",                    /*half narrow space*/
"\001 ",                     /*unpaddable space*/
"\001\016A\017",             /*alpha*/
"\001\016B\017",             /*beta*/
"\001\016C\017",             /*gamma*/
"\001\016D\017",             /*delta*/
"\001\016E\017",             /*epsilon*/
"\001\016F\017",             /*zeta*/
"\001\016G\017",             /*eta*/
"\001\016H\017",             /*theta*/
"\001\016I\017",             /*iota*/
"\001\016J\017",             /*kappa*/
"\001\016K\017",             /*lambda*/
"\001\016L\017",             /*mu*/
"\001\016M\017",             /*nu*/
"\001\016N\017",             /*xi*/
"\001\016O\017",             /*omicron*/
"\001\016P\017",             /*pi*/
"\001\016Q\017",             /*rho*/
"\001\016R\017",             /*sigma*/
"\001\016S\017",             /*tau*/
"\001\016T\017",             /*upsilon*/
"\001\016U\017",             /*phi*/
"\001\016V\017",             /*chi*/
"\001\016W\017",             /*psi*/
"\001\016X\017",             /*omega*/
"\001\016#\017",             /*Gamma*/
"\001\016$\017",             /*Delta*/
"\001\016(\017",             /*Theta*/
"\001\016+\017",             /*Lambda*/
"\001\016.\017",             /*Xi*/
"\001\0160\017",             /*Pi*/
"\001\0169\017",             /*Sigma*/
"\000",                      /**/
"\001\0164\017",             /*Upsilon*/
"\001\0165\017",             /*Phi*/
"\001\0167\017",             /*Psi*/
"\001\0168\017",             /*Omega*/
"\001\016[\017",             /*square root*/
"\001\016Y\017",             /*\(ts yields script-l*/
"\001\016k\017",             /*root en*/
"\001>\b_",                  /*>=*/
"\001<\b_",                  /*<=*/
"\001=\b_",                  /*identically equal*/
"\001-",                     /*equation minus*/
"\001\016o\017",             /*approx =*/
```

```
"\001\016n\017",                                    /*approximates*/
"\001=\b/",                                         /*not equal*/
"\002-\242-\202>",                                  /*right arrow*/
"\002<\b\202-\242\200-",                            /*left arrow*/
"\001|\b^",                                         /*up arrow*/
"\001|\b\302v\342",                                 /*down arrow*/
"\001=",                                            /*equation equal*/
"\001\016|\017",                                    /*multiply*/
"\001\016}\017",                                    /*divide*/
"\001\016j\017",                                    /*plus-minus*/
"\001\243|\203_\203|\243",                          /*cup (union)*/
"\001\243|\203\351_\311\203|\243",                  /*cap (intersection)*/
"\001\243(\203\302-\345-\303",                      /*subset of*/
"\001\302-\345-\303\203)\243",                      /*superset of*/
"\001_\b\243(\203\302-\345-\303",                   /*improper subset*/
"\001_\b\302-\345-\303\203)\243",                   /*improper superset*/
"\001\016~\017",                                    /*infinity*/
"\001\200o\201\301'\241\341'\241\341'\201\301",     /*partial derivative*/
"\001\016:\017",                                    /*gradient*/
"\001\200-\202\341,\301\242",                       /*not*/
"\001\016?\017",                                    /*integral sign*/
"\002o\242c\202",                                   /*proportional to*/
"\001O\b/",                                         /*empty set*/
"\001<\b\341-\302",                                 /*member of*/
"\001+",                                            /*equation plus*/
"\003(R)",                                          /*registered*/
"\003(C)",                                          /*copyright*/
"\001|",                                            /*box rule */
"\001\033Y",                                        /*cent sign*/
"\001|\b\342=\302",                                 /*double dagger*/
"\002=>",                                           /*right hand*/
"\002<=",                                           /*left hand*/
"\001*",                                            /*math * */
"\001\0162\017",                                    /*\(bs yields small sigma*/
"\001|",                                            /*or (was star)*/
"\001O",                                            /*circle*/
"\001|",                                            /*left top of big brace*/
"\001|",                                            /*left bot of big brace*/
"\001|",                                            /*right top of big brace*/
"\001|",                                            /*right bot of big brace*/
"\001\016]\017",                                    /*left center of big brace*/
"\001\016\\\017",                                   /*right center of big brace*/
"\001|",                                            /*bold vertical*/
"\001|",                                            /*left floor (lb of big bracket)*/
"\001|",                                            /*right floor (rb of big bracket)*/
"\001|",                                            /*left ceiling (lt of big bracket)*/
"\001|"                                             /*right ceiling (rt of big bracket)*/
```

FILES

/usr/lib/term/tab*name*	driving tables
/usr/lib/term/README	list of terminals supported by **nroff**(1)

SEE ALSO

nroff(1), **ttcompat**(4M)

NAME

term – format of compiled term file

SYNOPSIS

term

DESCRIPTION

Compiled **terminfo** descriptions are placed under the directory **/usr/share/lib/terminfo**. In order to avoid a linear search of a huge system directory, a two-level scheme is used: **/usr/share/lib/terminfo/c/name** where *name* is the name of the terminal, and *c* is the first character of *name*. Thus, *act4* can be found in the file **/usr/share/lib/terminfo/a/act4**. Synonyms for the same terminal are implemented by multiple links to the same compiled file.

The format has been chosen so that it will be the same on all hardware. An 8 or more bit byte is assumed, but no assumptions about byte ordering or sign extension are made.

The compiled file is created with the **tic**(8V) program, and read by the routine **setupterm** (see **curses**(3V)). Both of these pieces of software are part of **curses**(3V). The file is divided into six parts:

the header,
terminal names,
boolean flags,
numbers,
strings,
and
string table.

The header section begins the file. This section contains six short integers in the format described below. These integers are:

(1) the magic number (octal 0432);
(2) the size, in bytes, of the names section;
(3) the number of bytes in the boolean section;
(4) the number of short integers in the numbers section;
(5) the number of offsets (short integers) in the strings section;
(6) the size, in bytes, of the string table.

Short integers are stored in two 8-bit bytes. The first byte contains the least significant 8 bits of the value, and the second byte contains the most significant 8 bits. (Thus, the value represented is 256*second+first.) The value −1 is represented by 0377, 0377, other negative value are illegal. The −1 generally means that a capability is missing from this terminal. Note: this format corresponds to the hardware of the VAX and PDP-11. Machines where this does not correspond to the hardware read the integers as two bytes and compute the result.

The terminal names section comes next. It contains the first line of the terminfo description, listing the various names for the terminal, separated by the '|' character. The section is terminated with an ASCII NUL character.

The boolean flags have one byte for each flag. This byte is either 0 or 1 as the flag is present or absent. The capabilities are in the same order as the file **<term.h>**.

Between the boolean section and the number section, a null byte will be inserted, if necessary, to ensure that the number section begins on an even byte. All short integers are aligned on a short word boundary.

The numbers section is similar to the flags section. Each capability takes up two bytes, and is stored as a short integer. If the value represented is −1, the capability is taken to be missing.

The strings section is also similar. Each capability is stored as a short integer, in the format above. A value of −1 means the capability is missing. Otherwise, the value is taken as an offset from the beginning of the string table. Special characters in ^X or \c notation are stored in their interpreted form, not the printing representation. Padding information **$<*nn*>** and parameter information *%x* are stored intact in uninterpreted form.

The final section is the string table. It contains all the values of string capabilities referenced in the string section. Each string is null-terminated.

Note: it is possible for **setupterm** to expect a different set of capabilities than are actually present in the file. Either the database may have been updated since **setupterm** has been recompiled (resulting in extra unrecognized entries in the file) or the program may have been recompiled more recently than the database was updated (resulting in missing entries). The routine **setupterm** must be prepared for both possibilities — this is why the numbers and sizes are included. Also, new capabilities must always be added at the end of the lists of boolean, number, and string capabilities.

As an example, an octal dump of the description for the Microterm ACT 4 is included:

```
microterm|act4|microterm act iv,
cr=^M, cud1=^J, ind=^J, bel=^G, am, cub1=^H,
ed=^_, el=^^, clear=^L, cup=^T%p1%c%p2%c,
cols#80, lines#24, cuf1=^X, cuu1=^Z, home=^],

000 032 001   \0 025  \0  \b  \0 212  \0   "  \0   m   i   c   r
020   o   t   e   r   m   |   a   c   t   4   |   m   i   c   r   o
040   t   e   r   m       a   c   t       i   v  \0  \0 001  \0  \0
060  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
100  \0  \0   P  \0 377 377 030  \0 377 377 377 377 377 377 377 377
120 377 377 377 377  \0  \0 002  \0 377 377 377 377 004  \0 006  \0
140  \b  \0 377 377 377 377  \n  \0 026  \0 030  \0 377 377 032  \0
160 377 377 377 377 034  \0 377 377 036  \0 377 377 377 377 377 377
200 377 377 377 377 377 377 377 377 377 377 377 377 377 377 377 377
*
520 377 377 377 377       \0 377 377 377 377 377 377 377 377 377 377
540 377 377 377 377 377 377 007  \0  \r  \0  \f  \0 036  \0 037  \0
560 024   %   p   1   %   c   %   p   2   %   c  \0  \n  \0 035  \0
600  \b  \0 030  \0 032  \0  \n  \0
```

Some limitations: total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

FILES

/usr/share/lib/terminfo/*/*
compiled terminal capability data base

SEE ALSO

curses(3V), **terminfo**(5V), **tic**(8V)

NAME

termcap – terminal capability data base

DESCRIPTION

termcap is a data base describing the capabilities of terminals. Terminals are described in **termcap** source descriptions by giving a set of capabilities which they have, by describing how operations are performed, by describing padding requirements, and by specifying initialization sequences. This database is used by applications programs such as **vi**(1), and libraries such as **curses**(3V), so they can work with a variety of terminals without changes to the programs.

Each **termcap** entry consist of a number of colon-separated (:) fields. The first field for each terminal lists the various names by which it is known, separated by bar (|) characters. The first name is always two characters long, and is used by older (version 6) systems (which store the terminal type in a 16-bit word in a system-wide database). The second name given is the most common abbreviation for the terminal (this is the one to which the environment variable **TERM** would normally be set). The last name should fully identify the terminal's make and model. All other names are taken as synonyms for the initial terminal name. All names but the first and last should be in lower case and contain no blanks; the last name may well contain upper case and blanks for added readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions:

- The particular piece of hardware making up the terminal should have a root name chosen; for example, for the Hewlett-Packard 2621, **hp2621**. This name should not contain hyphens.
- Modes that the hardware can be in or user preferences should be indicated by appending a hyphen and an indicator of the mode. Thus, a **vt100** in 132-column mode would be given as: **vt100–w**. The following suffixes should be used where possible:

Suffix	*Meaning*	*Example*
–w	wide mode (more than 80 columns)	**vt100–w**
–am	with automatic margins (usually default)	**vt100–am**
–nam	without automatic margins	**vt100–nam**
–n	number of lines on the screen	**aaa–60**
–na	no arrow keys (leave them in local)	**concept100–na**
*–n***p**	number of pages of memory	**concept100–4p**
–rv	reverse video	**concept100–rv**

Terminal entries may continue onto multiple lines by giving a \ as the last character of a line, and empty fields may be included for readability (here between the last field on a line and the first field on the next). Comments may be included on lines beginning with #.

Types of Capabilities

Terminal capabilities each have a two-letter code, and are of three types:

boolean These indicate particular features of the terminal. For instance, an entry for a terminal that has automatic margins (an automatic RETURN and LINEFEED when the end of a line is reached) would contain a field with the boolean capability **am**.

numeric These give the size of the display of some other attribute. Numeric capabilities are followed by the character '#', and a number. An entry for a teminal with an 80-column display would have a field containing **co#80**.

string These indicate the character sequences used to perform particular terminal operations. String-valued capabilities, such as **ce** (clear-to-end-of-line sequence) are given by the two-letter code, followed by the character '=', and a string (which ends at the following : field delimiter).

A delay factor, in milliseconds may appear after the '='. Padding characters are supplied by **tputs** after the remainder of the string is sent. The delay can be either a number, or a number followed by the character '*', which indicates that the proportional padding is required, in which case the number given is the

amount of padding for each line affected by an operation using that capability. (In the case of an insert-character operation, the factor is still the number of *lines* affected; this is always 1 unless the terminal has **in** and the software uses it.)

When a * is specified, it is sometimes useful to give a delay of the form **3.5** to specify a delay per line to tenths of milliseconds. (Only one decimal place is allowed.)

Comments

To comment-out a capability field, insert a '.' (period) as the first character in that field (following the :).

Escape Sequence Codes

A number of escape sequences are provided in the string-valued capabilities for easy encoding of characters there:

\E	maps to ESC
^X	maps to CTRL-*X* for any appropriate character *X*
\n	maps to LINEFEED
\r	maps to RETURN
\t	maps to TAB
\b	maps to BACKSPACE
\f	maps to FORMFEED

Finally, characters may be given as three octal digits after a backslash (for example, **\123**), and the characters ^ (caret) and \ (backslash) may be given as \^ and \\ respectively.

If it is necessary to place a : in a capability it must be escaped in octal as **\072**.

If it is necessary to place a NUL character in a string capability it must be encoded as **\200**. (The routines that deal with **termcap** use C strings and strip the high bits of the output very late, so that a **\200** comes out as a **\000** would.)

Parameterized Strings

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with **printf**(3V)-like escapes (*%x*) in it; other characters are passed through unchanged. For example, to address the cursor, the **cm** capability is given, using two parameters: the row and column to move to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory. If the terminal has memory-relative cursor addressing, that can be indicated by an analogous CM capability.)

The % escapes have the following meanings:

%%	produce the character %
%d	output *value* as in **printf %d**
%2	output *value* as in **printf %2d**
%3	output *value* as in **printf %3d**
%.	output *value* as in **printf %c**
%+*x*	add *x* to *value*, then do '**%.**'
%>*xy*	if *value* > *x* then add *y*, no output
%r	reverse order of two parameters, no output
%i	increment by one, no output
%n	exclusive-or all parameters with 0140 (Datamedia 2500)
%B	BCD (16*(*value*/10)) + (*value*%10), no output
%D	Reverse coding (*value* – 2*(*value*%16)), no output (Delta Data)

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent **\E&a12c03Y** padded for 6 milliseconds. Note: the order of the row and column coordinates is reversed here and that the row and column are sent as two-digit integers. Thus its **cm** capability is **':cm=6\E&%r%2c%2Y:'**. Terminals that use **'%.'** need to be able to backspace the cursor (**le**) and to move the cursor up one line on the screen (**up**). This is necessary because it is not always safe to transmit **\n**, **^D**, and **\r**, as the system may change or discard them. (Programs using **termcap** must set terminal modes so that TAB characters are not expanded, making **\t** safe to send. This turns out to be essential for the Ann Arbor 4080.)

A final example is the Lear Siegler ADM–3a, which offsets row and column by a blank character, thus it requires **':cm=\E=%+ %+:'**.

Row or column absolute cursor addressing can be given as single-parameter capabilities **ch** (horizontal position absolute) and **cv** (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to **cm**. If there are parameterized local motions (for example, move *n* positions to the right) these can be given as **DO**, **LE**, **RI**, and **UP** with a single parameter indicating how many positions to move. These are primarily useful if the terminal does not have **cm**, such as the Tektronix 4025.

Delays

Certain capabilities control padding in the terminal driver. These are primarily needed by hardcopy terminals and are used by the **tset (1)** program to set terminal driver modes appropriately. Delays embedded in the capabilities **cr**, **sf**, **le**, **ff**, and **ta** will set the appropriate delay bits in the terminal driver. If **pb** (padding baud rate) is given, these values can be ignored at baud rates below the value of **pb**. For 4.2BSD **tset**, the delays are given as numeric capabilities **dC**, **dN**, **dB**, **dF**, and **dT** instead.

Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability **tc** can be given with the name of the similar terminal. This capability must be *last*, and the combined length of the entries must not exceed 1024. The capabilities given before **tc** override those in the terminal type invoked by **tc**. A capability can be canceled by placing **xx@** to the left of the **tc** invocation, where *xx* is the capability. For example, the entry

hn|2621–nl:ks@:ke@:tc=2621:

defines a **2621–nl** that does not have the **ks** or **ke** capabilities, hence does not turn on the function key labels when in visual mode. This is useful for different modes for a terminal, or for different user preferences.

CAPABILITIES

The characters in the *Notes* field in the next table have the following meanings (more than one may apply to a capability):

- **N** indicates numeric parameter(s)
- **P** indicates that padding may be specified
- ***** indicates that padding may be based on the number of lines affected
- **o** indicates capability is obsolete

Obsolete capabilities have no **terminfo** equivalents, since they were considered useless, or are subsumed by other capabilities. New software should not rely on them.

Name	*Type*	*Notes*	*Description*
!1	*str*		sent by shifted save key
!2	*str*		sent by shifted suspend key
!3	*str*		sent by shifted undo key
#1	*str*		sent by shifted help key
#2	*str*		sent by shifted home key
#3	*str*		sent by shifted input key
#4	*str*		sent by shifted left-arrow key
%0	*str*		sent by redo key
%1	*str*		sent by help key

%2	*str*		sent by mark key
%3	*str*		sent by message key
%4	*str*		sent by move key
%5	*str*		sent by next-object key
%6	*str*		sent by open key
%7	*str*		sent by options key
%8	*str*		sent by previous-object key
%9	*str*		sent by print or copy key
%a	*str*		sent by shifted message key
%b	*str*		sent by shifted move key
%c	*str*		sent by shifted next-object key
%d	*str*		sent by shifted options key
%e	*str*		sent by shifted previous-object key
%f	*str*		sent by shifted print or copy key
%g	*str*		sent by shifted redo key
%h	*str*		sent by shifted replace key
%i	*str*		sent by shifted right-arrow key
%j	*str*		sent by shifted resume key
&0	*str*		sent by shifted cancel key
&1	*str*		sent by ref(erence) key
&2	*str*		sent by refresh key
&3	*str*		sent by replace key
&4	*str*		sent by restart key
&5	*str*		sent by resume key
&6	*str*		sent by save key
&7	*str*		sent by suspend key
&8	*str*		sent by undo key
&9	*str*		sent by shifted beg(inning) key
***0**	*str*		sent by shifted find key
***1**	*str*		sent by shifted cmd (command) key
***2**	*str*		sent by shifted copy key
***3**	*str*		sent by shifted create key
***4**	*str*		sent by shifted delete-char key
***5**	*str*		sent by shifted delete-line key
***6**	*str*		sent by select key
***7**	*str*		sent by shifted end key
***8**	*str*		sent by shifted clear-line key
***9**	*str*		sent by shifted exit key
5i	*bool*		printer will not echo on screen
@0	*str*		sent by find key
@1	*str*		sent by beg(inning) key
@2	*str*		sent by cancel key
@3	*str*		sent by close key
@4	*str*		sent by cmd (command) key
@5	*str*		sent by copy key
@6	*str*		sent by create key
@7	*str*		sent by end key
@8	*str*		sent by enter/send key (unreliable)
@9	*str*		sent by exit key
AL	*str*	*(NP*)*	add *n* new blank lines
CC	*str*		terminal settable command character in prototype
CM	*str*	*(NP)*	memory-relative cursor motion to row *m*, column *n*
DC	*str*	*(NP*)*	delete *n* characters
DL	*str*	*(NP*)*	delete *n* lines
DO	*str*	*(NP*)*	move cursor down *n* lines
EP	*bool*	*(o)*	even parity
F1-F9	*str*		sent by function keys 11-19
FA-FZ	*str*		sent by function keys 20-45

Fa-Fr	*str*		sent by function keys 46-63
HC	*bool*		cursor is hard to see
HD	*bool*	*(o)*	half-duplex
IC	*str*	*(NP*)*	insert *n* blank characters
K1	*str*		sent by keypad upper left
K2	*str*		sent by keypad center
K3	*str*		sent by keypad upper right
K4	*str*		sent by keypad lower left
K5	*str*		sent by keypad lower right
LC	*bool*	*(o)*	lower-case only
LE	*str*	*(NP)*	move cursor left *n* positions
LF	*str*	*(P)*	turn off soft labels
LO	*str*	*(P)*	turn on soft labels
MC	*str*	*(P)*	clear left and right soft margins
ML	*str*	*(P)*	set soft left margin
MR	*str*	*(P)*	set soft right margin
NL	*bool*	*(o)*	**\n** is NEWLINE, not LINEFEED
NP	*bool*		pad character does not exist
NR	*bool*		**ti** does not reverse **te**
Nl	*num*		number of labels on screen (start at 1)
OP	*bool*	*(o)*	odd parity
RA	*str*	*(P)*	turn off automatic margins
RF	*str*		send next input character (for ptys)
RI	*str*	*(NP)*	move cursor right *n* positions
RX	*str*	*(P)*	turn off xoff/xon handshaking
SA	*str*	*(P)*	turn on automatic margins
SF	*str*	*(NP*)*	scroll forward *n* lines
SR	*str*	*(NP*)*	scroll backward *n* lines
SX	*str*	*(P)*	turn on xoff/xon handshaking
UC	*bool*	*(o)*	upper-case only
UP	*str*	*(NP*)*	move cursor up *n* lines
XF	*str*		x-off character (default DC3)
XN	*str*		x-on character (default DC1)
ac	*str*		graphic character set pairs aAbBcC – def=VT100
ae	*str*	*(P)*	end alternate character set
al	*str*	*(P*)*	add new blank line
am	*bool*		terminal has automatic margins
as	*str*	*(P)*	start alternate character set
bc	*str*	*(o)*	backspace if not **^H**
bl	*str*	*(P)*	audible signal (bell)
bs	*bool*	*(o)*	terminal can backspace with **^H**
bt	*str*	*(P)*	back-tab
bw	*bool*		**le** (backspace) wraps from column 0 to last column
cb	*str*	*(P)*	clear to beginning of line, inclusive
cd	*str*	*(P*)*	clear to end of display
ce	*str*	*(P)*	clear to end of line
ch	*str*	*(NP)*	set cursor column (horizontal position)
cl	*str*	*(P*)*	clear screen and home cursor
cm	*str*	*(NP)*	screen-relative cursor motion to row *m*, column *n*
co	*num*		number of columns in a line
cr	*str*	*(P*)*	RETURN
cs	*str*	*(NP)*	change scrolling region to lines *m* through *n* (VT100)
ct	*str*	*(P)*	clear all tab stops
cv	*str*	*(NP)*	set cursor row (vertical position)
dB	*num*	*(o)*	milliseconds of **bs** delay needed (default 0)
dC	*num*	*(o)*	milliseconds of **cr** delay needed (default 0)
dF	*num*	*(o)*	milliseconds of **ff** delay needed (default 0)
dN	*num*	*(o)*	milliseconds of **nl** delay needed (default 0)

dT	*num*	*(o)*	milliseconds of horizontal tab delay needed (default 0)
dV	*num*	*(o)*	milliseconds of vertical tab delay needed (default 0)
da	*bool*		display may be retained above the screen
db	*bool*		display may be retained below the screen
dc	*str*	*(P*)*	delete character
dl	*str*	*(P*)*	delete line
dm	*str*		enter delete mode
do	*str*		down one line
ds	*str*		disable status line
eA	*str*	*(P)*	enable graphic character set
ec	*str*	*(NP)*	erase *n* characters
ed	*str*		end delete mode
ei	*str*		end insert mode
eo	*bool*		can erase overstrikes with a blank
es	*bool*		escape can be used on the status line
ff	*str*	*(P*)*	hardcopy terminal page eject
fs	*str*		return from status line
gn	*bool*		generic line type (for example dialup, switch)
hc	*bool*		hardcopy terminal
hd	*str*		half-line down (forward 1/2 linefeed)
ho	*str*	*(P)*	home cursor
hs	*bool*		has extra "status line"
hu	*str*		half-line up (reverse 1/2 linefeed)
hz	*bool*		cannot print ~s (Hazeltine)
i1	*str*		terminal initialization string (**terminfo** only)
i3	*str*		terminal initialization string (**terminfo** only)
iP	*str*		pathname of program for initialization (**terminfo** only)
ic	*str*	*(P*)*	insert character
if	*str*		name of file containing initialization string
im	*str*		enter insert mode
in	*bool*		insert mode distinguishes nulls
ip	*str*	*(P*)*	insert pad after character inserted
is	*str*		terminal initialization string
it	*num*		tab stops initially every *n* positions
k0-k9	*str*		sent by function keys 0-9
k;	*str*		sent by function key 10
kA	*str*		sent by insert-line key
kB	*str*		sent by back-tab key
kC	*str*		sent by clear-screen or erase key
kD	*str*		sent by delete-character key
kE	*str*		sent by clear-to-end-of-line key
kF	*str*		sent by scroll-forward/down key
kH	*str*		sent by home-down key
kI	*str*		sent by insert-character or enter-insert-mode key
kL	*str*		sent by delete-line key
kM	*str*		sent by insert key while in insert mode
kN	*str*		sent by next-page key
kP	*str*		sent by previous-page key
kR	*str*		sent by scroll-backward/up key
kS	*str*		sent by clear-to-end-of-screen key
kT	*str*		sent by set-tab key
ka	*str*		sent by clear-all-tabs key
kb	*str*		sent by backspace key
kd	*str*		sent by down-arrow key
ke	*str*		out of "keypad transmit" mode
kh	*str*		sent by home key
kl	*str*		sent by left-arrow key
km	*bool*		has a "meta" key (shift, sets parity bit)

kn	*num*	*(o)*	number of function (**k0–k9**) keys (default 0)
ko	*str*	*(o)*	termcap entries for other non-function keys
kr	*str*		sent by right-arrow key
ks	*str*		put terminal in "keypad transmit" mode
kt	*str*		sent by clear-tab key
ku	*str*		sent by up-arrow key
l0-l9	*str*		labels on function keys 0-9 if not f0-f9
la	*str*		label on function key 10 if not f10
le	*str*	*(P)*	move cursor left one position
lh	*num*		number of rows in each label
li	*num*		number of lines on screen or page
ll	*str*		last line, first column
lm	*num*		lines of memory if > **li** (0 means varies)
lw	*num*		number of columns in each label
ma	*str*	*(o)*	arrow key map (used by *vi* version 2 only)
mb	*str*		turn on blinking attribute
md	*str*		turn on bold (extra bright) attribute
me	*str*		turn off all attributes
mh	*str*		turn on half-bright attribute
mi	*bool*		safe to move while in insert mode
mk	*str*		turn on blank attribute (characters invisible)
ml	*str*	*(o)*	memory lock on above cursor
mm	*str*		turn on "meta mode" (8th bit)
mo	*str*		turn off "meta mode"
mp	*str*		turn on protected attribute
mr	*str*		turn on reverse-video attribute
ms	*bool*		safe to move in standout modes
mu	*str*	*(o)*	memory unlock (turn off memory lock)
nc	*bool*	*(o)*	no correctly-working **cr** (Datamedia 2500, Hazeltine 2000)
nd	*str*		non-destructive space (cursor right)
nl	*str*	*(o)*	NEWLINE character if not
ns	*bool*	*(o)*	terminal is a CRT but does not scroll
nw	*str*	*(P)*	NEWLINE (behaves like **cr** followed by **do**)
nx	*bool*		padding will not work, xoff/xon required
os	*bool*		terminal overstrikes
pO	*str*	*(N)*	turn on the printer for *n* bytes
pb	*num*		lowest baud where delays are required
pc	*str*		pad character (default NUL)
pf	*str*		turn off the printer
pk	*str*		program function key *n* to type string *s* (**terminfo** only)
pl	*str*		program function key *n* to execute string *s* (**terminfo** only)
pn	*str*	*(NP)*	program label *n* to show string *s* (**terminfo** only)
po	*str*		turn on the printer
ps	*str*		print contents of the screen
pt	*bool*	*(o)*	has hardware tab stops (may need to be set with **is**)
px	*str*		program function key *n* to transmit string *s* (**terminfo** only)
r1	*str*		reset terminal completely to sane modes (**terminfo** only)
r2	*str*		reset terminal completely to sane modes (**terminfo** only)
r3	*str*		reset terminal completely to sane modes (**terminfo** only)
rP	*str*	*(P)*	like **ip** but when in replace mode
rc	*str*	*(P)*	restore cursor to position of last **sc**
rf	*str*		name of file containing reset string
ri	*?*		unknown at present
rp	*str*	*(NP*)*	repeat character *c* *n* times
rs	*str*		reset terminal completely to sane modes
sa	*str*	*(NP)*	define the video attributes (9 parameters)
sc	*str*	*(P)*	save cursor position
se	*str*		end standout mode

sf	*str*	*(P)*	scroll text up
sg	*num*		number of garbage chars left by **so** or **se** (default 0)
so	*str*		begin standout mode
sr	*str*	*(P)*	scroll text down
st	*str*		set a tab stop in all rows, current column
ta	*str*	*(P)*	move cursor to next 8-position hardware tab stop
tc	*str*		entry of similar terminal – must be last
te	*str*		string to end programs that use **termcap**
ti	*str*		string to begin programs that use **termcap**
ts	*str*	*(N)*	go to status line, column *n*
uc	*str*		underscore one character and move past it
ue	*str*		end underscore mode
ug	*num*		number of garbage chars left by **us** or **ue** (default 0)
ul	*bool*		underline character overstrikes
up	*str*		upline (cursor up)
us	*str*		start underscore mode
vb	*str*		visible bell (must not move cursor)
ve	*str*		make cursor appear normal (undo **vs**/**vi**)
vi	*str*		make cursor invisible
vs	*str*		make cursor very visible
vt	*num*		virtual terminal number (not supported on all systems)
wi	*str*	*(N)*	set current window to lines *i* through *j*, columns *m* through *n*
ws	*num*		number of columns in status line
xb	*bool*		Beehive (f1=ESC, f2=^C)
xn	*bool*		NEWLINE ignored after 80 cols (Concept)
xo	*bool*		terminal uses xoff/xon handshaking
xr	*bool*	*(o)*	RETURN acts like **ce cr nl** (Delta Data)
xs	*bool*		standout not erased by overwriting (Hewlett-Packard)
xt	*bool*		TAB characters destructive, magic **so** char (Teleray 1061)
xx	*bool*	*(o)*	Tektronix 4025 insert-line

ENVIRONMENT

If the environment variable **TERMCAP** contains an absolute pathname, programs look to that file for terminal descriptions, rather than **/usr/share/lib/termcap**. If the value of this variable is in the form of a **termcap** entry, programs use that value for the terminal description.

FILES

/usr/share/lib/termcap file containing terminal descriptions

SEE ALSO

ex(1), **more**(1), **tset**(1), **ul**(1), **vi**(1), **curses**(3V), **printf**(3V), **termcap**(3X), **term**(5V), **terminfo**(5V)

System and Network Administration

WARNINGS

UNIX System V uses **terminfo**(5V) rather than **termcap**. SunOS supports either **termcap** or **terminfo**(5V) terminal databases, depending on whether you link with the **termcap**(3X) or **curses**(3V) libraries. Transitions between the two should be relatively painless if capabilities flagged as "obsolete" are avoided.

vi allows only 256 characters for string capabilities, and the routines in **termcap**(3X) do not check for overflow of this buffer. The total length of a single entry (excluding only escaped NEWLINE characters) may not exceed 1024.

Not all programs support all entries.

NAME

terminfo – terminal capability data base

SYNOPSIS

/usr/share/lib/terminfo/?/*

AVAILABILITY

This database is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

terminfo is a compiled database (see **tic**(8V)) describing the capabilities of terminals. Terminals are described in **terminfo** source descriptions by giving a set of capabilities which they have, by describing how operations are performed, by describing padding requirements, and by specifying initialization sequences. This database is used by applications programs, and by libraries such as **curses**(3V), so they can work with a variety of terminals without changes to the programs. To obtain the source description for a terminal, use the **–I** option of **infocmp**(8V).

Entries in **terminfo** source files consist of a number of comma-separated fields. White space after each comma is ignored. The first line of each terminal description in the **terminfo** database gives the name by which **terminfo** knows the terminal, separated by pipe (|) characters. The first name given is the most common abbreviation for the terminal (this is the one to which the environment variable **TERM** would normally be set), the last name given should be a long name fully identifying the terminal, and all others are understood as synonyms for the terminal name. All names but the last should contain no blanks; the last name may contain blanks for readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions:

- The particular piece of hardware making up the terminal should have a root name chosen; for example, for the Hewlett-Packard 2621, **hp2621**. This name should not contain hyphens.
- Modes that the hardware can be in or user preferences should be indicated by appending a hyphen and an indicator of the mode. Thus, a **vt100** in 132-column mode would be given as: **vt100–w**. The following suffixes should be used where possible:

Suffix	*Meaning*	*Example*
–w	wide mode (more than 80 columns)	**vt100–w**
–am	with automatic margins (usually default)	**vt100–am**
–nam	without automatic margins	**vt100–nam**
–n	number of lines on the screen	**aaa–60**
–na	no arrow keys (leave them in local)	**concept100–na**
*–n***p**	number of pages of memory	**concept100–4p**
–rv	reverse video	**concept100–rv**

CAPABILITIES

In the table below, the **Variable** is the name by which the C programmer (at the **terminfo** level) accesses the capability. The **capname** is the short name for this variable used in the text of the database. It is used by a person updating the database and by the **tput**(1V) command when asking what the value of the capability is for a particular terminal. The **Termcap Code** is a two-letter code that corresponds to the old **termcap** capability name.

Capability names have no hard length limit, but an informal limit of 5 characters has been adopted to keep them short. Whenever possible, names are chosen to be the same as or similar to the ANSI X3.64-1979 standard. Semantics are also intended to match those of the specification.

All string capabilities listed below may have padding specified, with the exception of those used for input. Input capabilities, listed under the **Strings** section in the table below, have names beginning with '**key_**'. The following indicators may appear at the end of the **Description** for a variable.

(G) indicates that the string is passed through **tparm**( ) with parameters (parms) as given ($\#_i$).
(*) indicates that padding may be based on the number of lines affected.
($\#_i$) indicates the i^{th} parameter.

Variable	*Capname*	*Termcap*	*Description*
Boolean			
auto_left_margin	**bw**	**bw**	**cub**1 wraps from column 0 to last column
auto_right_margin	**am**	**am**	Terminal has automatic margins
no_esc_ctlc	**xsb**	**xb**	Beehive (f1=ESC, f2=^C)
ceol_standout_glitch	**xhp**	**xs**	Standout not erased by overwriting (Hewlett-Packard)
eat_newline_glitch	**xenl**	**xn**	NEWLINE ignored after 80 cols (Concept)
erase_overstrike	**eo**	**eo**	Can erase overstrikes with a blank
generic_type	**gn**	**gn**	Generic line type (for example, dialup, switch).
hard_copy	**hc**	**hc**	Hardcopy terminal
hard_cursor	**chts**	**HC**	Cursor is hard to see
has_meta_key	**km**	**km**	Has a meta key (shift, sets parity bit)
has_status_line	**hs**	**hs**	Has extra "status line"
insert_null_glitch	**in**	**in**	Insert mode distinguishes nulls
memory_above	**da**	**da**	Display may be retained above the screen
memory_below	**db**	**db**	Display may be retained below the screen
move_insert_mode	**mir**	**mi**	Safe to move while in insert mode
move_standout_mode	**msgr**	**ms**	Safe to move in standout modes
needs_xon_xoff	**nxon**	**nx**	Padding will not work, xon/xoff required
non_rev_rmcup	**nrrmc**	**NR**	**smcup** does not reverse **rmcup**
no_pad_char	**npc**	**NP**	Pad character does not exist
over_strike	**os**	**os**	Terminal overstrikes on hard-copy terminal
prtr_silent	**mc5i**	**5i**	Printer will not echo on screen
status_line_esc_ok	**eslok**	**es**	Escape can be used on the status line
dest_tabs_magic_smso	**xt**	**xt**	Destructive TAB characters, magic **smso** char (Teleray 1061)
tilde_glitch	**hz**	**hz**	Hazeltine; cannot print tildes(~)
transparent_underline	**ul**	**ul**	Underline character overstrikes
xon_xoff	**xon**	**xo**	Terminal uses xon/xoff handshaking
Number			
columns	**cols**	**co**	Number of columns in a line
init_tabs	**it**	**it**	tab stops initially every # spaces
label_height	**lh**	**lh**	Number of rows in each label
label_width	**lw**	**lw**	Number of cols in each label
lines	**lines**	**li**	Number of lines on screen or page
lines_of_memory	**lm**	**lm**	Lines of memory if > **lines**; **0** means varies
magic_cookie_glitch	**xmc**	**sg**	Number blank chars left by **smso** or **rmso**
num_labels	**nlab**	**Nl**	Number of labels on screen (start at 1)
padding_baud_rate	**pb**	**pb**	Lowest baud rate where padding needed
virtual_terminal	**vt**	**vt**	Virtual terminal number (not supported on all systems)
width_status_line	**wsl**	**ws**	Number of columns in status line
String			
acs_chars	**acsc**	**ac**	Graphic charset pairs aAbBcC - def=VT100
back_tab	**cbt**	**bt**	Back tab
bell	**bel**	**bl**	Audible signal (bell)
carriage_return	**cr**	**cr**	RETURN (*)
change_scroll_region	**csr**	**cs**	Change to lines #1 through #2 (VT100) (G)

char_padding	**rmp**	**rP**	Like **ip** but when in replace mode
clear_all_tabs	**tbc**	**ct**	Clear all tab stops
clear_margins	**mgc**	**MC**	Clear left and right soft margins
clear_screen	**clear**	**cl**	Clear screen and home cursor (*)
clr_bol	**el1**	**cb**	Clear to beginning of line, inclusive
clr_eol	**el**	**ce**	Clear to end of line
clr_eos	**ed**	**cd**	Clear to end of display (*)
column_address	**hpa**	**ch**	Horizontal position absolute (G)
command_character	**cmdch**	**CC**	Terminal settable command char in prototype
cursor_address	**cup**	**cm**	Cursor motion to row #1 col #2 (G)
cursor_down	**cud1**	**do**	Down one line
cursor_home	**home**	**ho**	Home cursor (if no **cup**)
cursor_invisible	**civis**	**vi**	Make cursor invisible
cursor_left	**cub1**	**le**	Move cursor left one SPACE
cursor_mem_address	**mrcup**	**CM**	Memory relative cursor addressing (G)
cursor_normal	**cnorm**	**ve**	Make cursor appear normal (undo **cvvis/civis**)
cursor_right	**cuf1**	**nd**	Non-destructive space (cursor right)
cursor_to_ll	**ll**	**ll**	Last line, first column (if no **cup**)
cursor_up	**cuu1**	**up**	Upline (cursor up)
cursor_visible	**cvvis**	**vs**	Make cursor very visible
delete_character	**dch1**	**dc**	Delete character (*)
delete_line	**dl1**	**dl**	Delete line (*)
dis_status_line	**dsl**	**ds**	Disable status line
down_half_line	**hd**	**hd**	Half-line down (forward 1/2 LINEFEED)
ena_acs	**enacs**	**eA**	Enable alternate char set
enter_alt_charset_mode	**smacs**	**as**	Start alternate character set
enter_am_mode	**smam**	**SA**	Turn on automatic margins
enter_blink_mode	**blink**	**mb**	Turn on blinking
enter_bold_mode	**bold**	**md**	Turn on bold (extra bright) mode
enter_ca_mode	**smcup**	**ti**	String to begin programs that use **cup**
enter_delete_mode	**smdc**	**dm**	Delete mode (enter)
enter_dim_mode	**dim**	**mh**	Turn on half-bright mode
enter_insert_mode	**smir**	**im**	Insert mode (enter);
enter_protected_mode	**prot**	**mp**	Turn on protected mode
enter_reverse_mode	**rev**	**mr**	Turn on reverse video mode
enter_secure_mode	**invis**	**mk**	Turn on blank mode (chars invisible)
enter_standout_mode	**smso**	**so**	Begin standout mode
enter_underline_mode	**smul**	**us**	Start underscore mode
enter_xon_mode	**smxon**	**SX**	Turn on xon/xoff handshaking
erase_chars	**ech**	**ec**	Erase #1 characters (G)
exit_alt_charset_mode	**rmacs**	**ae**	End alternate character set
exit_am_mode	**rmam**	**RA**	Turn off automatic margins
exit_attribute_mode	**sgr0**	**me**	Turn off all attributes
exit_ca_mode	**rmcup**	**te**	String to end programs that use **cup**
exit_delete_mode	**rmdc**	**ed**	End delete mode
exit_insert_mode	**rmir**	**ei**	End insert mode;
exit_standout_mode	**rmso**	**se**	End standout mode
exit_underline_mode	**rmul**	**ue**	End underscore mode
exit_xon_mode	**rmxon**	**RX**	Turn off xon/xoff handshaking
flash_screen	**flash**	**vb**	Visible bell (must not move cursor)
form_feed	**ff**	**ff**	Hardcopy terminal page eject (*)
from_status_line	**fsl**	**fs**	Return from status line
init_1string	**is1**	**i1**	Terminal initialization string
init_2string	**is2**	**is**	Terminal initialization string
init_3string	**is3**	**i3**	Terminal initialization string
init_file	**if**	**if**	Name of initialization file containing **is**
init_prog	**iprog**	**iP**	Path name of program for init
insert_character	**ich1**	**ic**	Insert character

insert_line	**il1**	**al**	Add new blank line (*)
insert_padding	**ip**	**ip**	Insert pad after character inserted (*)
key_a1	**ka1**	**K1**	KEY_A1, 0534, Upper left of keypad
key_a3	**ka3**	**K3**	KEY_A3, 0535, Upper right of keypad
key_b2	**kb2**	**K2**	KEY_B2, 0536, Center of keypad
key_backspace	**kbs**	**kb**	KEY_BACKSPACE, 0407, Sent by BACKSPACE key
key_beg	**kbeg**	**@1**	KEY_BEG, 0542, Sent by beg(inning) key
key_btab	**kcbt**	**kB**	KEY_BTAB, 0541, Sent by back-tab key
key_c1	**kc1**	**K4**	KEY_C1, 0537, Lower left of keypad
key_c3	**kc3**	**K5**	KEY_C3, 0540, Lower right of keypad
key_cancel	**kcan**	**@2**	KEY_CANCEL, 0543, Sent by cancel key
key_catab	**ktbc**	**ka**	KEY_CATAB, 0526, Sent by clear-all-tabs key
key_clear	**kclr**	**kC**	KEY_CLEAR, 0515, Sent by clear- screen or erase key
key_close	**kclo**	**@3**	KEY_CLOSE, 0544, Sent by close key
key_command	**kcmd**	**@4**	KEY_COMMAND, 0545, Sent by cmd (command) key
key_copy	**kcpy**	**@5**	KEY_COPY, 0546, Sent by copy key
key_create	**kcrt**	**@6**	KEY_CREATE, 0547, Sent by create key
key_ctab	**kctab**	**kt**	KEY_CTAB, 0525, Sent by clear-tab key
key_dc	**kdch1**	**kD**	KEY_DC, 0512, Sent by delete-character key
key_dl	**kdl1**	**kL**	KEY_DL, 0510, Sent by delete-line key
key_down	**kcud1**	**kd**	KEY_DOWN, 0402, Sent by terminal down-arrow key
key_eic	**krmir**	**kM**	KEY_EIC, 0514, Sent by **rmir** or **smir** in insert mode
key_end	**kend**	**@7**	KEY_END, 0550, Sent by end key
key_enter	**kent**	**@8**	KEY_ENTER, 0527, Sent by enter/send key
key_eol	**kel**	**kE**	KEY_EOL, 0517, Sent by clear-to-end- of-line key
key_eos	**ked**	**kS**	KEY_EOS, 0516, Sent by clear-to-end- of-screen key
key_exit	**kext**	**@9**	KEY_EXIT, 0551, Sent by exit key
key_f0	**kf0**	**k0**	KEY_F(0), 0410, Sent by function key f0
key_f1	**kf1**	**k1**	KEY_F(1), 0411, Sent by function key f1
key_f2	**kf2**	**k2**	KEY_F(2), 0412, Sent by function key f2
key_f3	**kf3**	**k3**	KEY_F(3), 0413, Sent by function key f3
key_f4	**kf4**	**k4**	KEY_F(4), 0414, Sent by function key f4
key_f5	**kf5**	**k5**	KEY_F(5), 0415, Sent by function key f5
key_f6	**kf6**	**k6**	KEY_F(6), 0416, Sent by function key f6
key_f7	**kf7**	**k7**	KEY_F(7), 0417, Sent by function key f7
key_f8	**kf8**	**k8**	KEY_F(8), 0420, Sent by function key f8
key_f9	**kf9**	**k9**	KEY_F(9), 0421, Sent by function key f9
key_f10	**kf10**	**k;**	KEY_F(10), 0422, Sent by function key f10
key_f11	**kf11**	**F1**	KEY_F(11), 0423, Sent by function key f11
key_f12	**kf12**	**F2**	KEY_F(12), 0424, Sent by function key f12
key_f13	**kf13**	**F3**	KEY_F(13), 0425, Sent by function key f13
key_f14	**kf14**	**F4**	KEY_F(14), 0426, Sent by function key f14
key_f15	**kf15**	**F5**	KEY_F(15), 0427, Sent by function key f15
key_f16	**kf16**	**F6**	KEY_F(16), 0430, Sent by function key f16
key_f17	**kf17**	**F7**	KEY_F(17), 0431, Sent by function key f17
key_f18	**kf18**	**F8**	KEY_F(18), 0432, Sent by function key f18
key_f19	**kf19**	**F9**	KEY_F(19), 0433, Sent by function key f19
key_f20	**kf20**	**FA**	KEY_F(20), 0434, Sent by function key f20
key_f21	**kf21**	**FB**	KEY_F(21), 0435, Sent by function key f21
key_f22	**kf22**	**FC**	KEY_F(22), 0436, Sent by function key f22
key_f23	**kf23**	**FD**	KEY_F(23), 0437, Sent by function key f23
key_f24	**kf24**	**FE**	KEY_F(24), 0440, Sent by function key f24
key_f25	**kf25**	**FF**	KEY_F(25), 0441, Sent by function key f25
key_f26	**kf26**	**FG**	KEY_F(26), 0442, Sent by function key f26
key_f27	**kf27**	**FH**	KEY_F(27), 0443, Sent by function key f27
key_f28	**kf28**	**FI**	KEY_F(28), 0444, Sent by function key f28
key_f29	**kf29**	**FJ**	KEY_F(29), 0445, Sent by function key f29
key_f30	**kf30**	**FK**	KEY_F(30), 0446, Sent by function key f30

key_f31	**kf31**	**FL**	KEY_F(31), 0447, Sent by function key f31
key_f32	**kf32**	**FM**	KEY_F(32), 0450, Sent by function key f32
key_f33	**kf33**	**FN**	KEY_F(13), 0451, Sent by function key f13
key_f34	**kf34**	**FO**	KEY_F(34), 0452, Sent by function key f34
key_f35	**kf35**	**FP**	KEY_F(35), 0453, Sent by function key f35
key_f36	**kf36**	**FQ**	KEY_F(36), 0454, Sent by function key f36
key_f37	**kf37**	**FR**	KEY_F(37), 0455, Sent by function key f37
key_f38	**kf38**	**FS**	KEY_F(38), 0456, Sent by function key f38
key_f39	**kf39**	**FT**	KEY_F(39), 0457, Sent by function key f39
key_f40	**kf40**	**FU**	KEY_F(40), 0460, Sent by function key f40
key_f41	**kf41**	**FV**	KEY_F(41), 0461, Sent by function key f41
key_f42	**kf42**	**FW**	KEY_F(42), 0462, Sent by function key f42
key_f43	**kf43**	**FX**	KEY_F(43), 0463, Sent by function key f43
key_f44	**kf44**	**FY**	KEY_F(44), 0464, Sent by function key f44
key_f45	**kf45**	**FZ**	KEY_F(45), 0465, Sent by function key f45
key_f46	**kf46**	**Fa**	KEY_F(46), 0466, Sent by function key f46
key_f47	**kf47**	**Fb**	KEY_F(47), 0467, Sent by function key f47
key_f48	**kf48**	**Fc**	KEY_F(48), 0470, Sent by function key f48
key_f49	**kf49**	**Fd**	KEY_F(49), 0471, Sent by function key f49
key_f50	**kf50**	**Fe**	KEY_F(50), 0472, Sent by function key f50
key_f51	**kf51**	**Ff**	KEY_F(51), 0473, Sent by function key f51
key_f52	**kf52**	**Fg**	KEY_F(52), 0474, Sent by function key f52
key_f53	**kf53**	**Fh**	KEY_F(53), 0475, Sent by function key f53
key_f54	**kf54**	**Fi**	KEY_F(54), 0476, Sent by function key f54
key_f55	**kf55**	**Fj**	KEY_F(55), 0477, Sent by function key f55
key_f56	**kf56**	**Fk**	KEY_F(56), 0500, Sent by function key f56
key_f57	**kf57**	**Fl**	KEY_F(57), 0501, Sent by function key f57
key_f58	**kf58**	**Fm**	KEY_F(58), 0502, Sent by function key f58
key_f59	**kf59**	**Fn**	KEY_F(59), 0503, Sent by function key f59
key_f60	**kf60**	**Fo**	KEY_F(60), 0504, Sent by function key f60
key_f61	**kf61**	**Fp**	KEY_F(61), 0505, Sent by function key f61
key_f62	**kf62**	**Fq**	KEY_F(62), 0506, Sent by function key f62
key_f63	**kf63**	**Fr**	KEY_F(63), 0507, Sent by function key f63
key_find	**kfnd**	**@0**	KEY_FIND, 0552, Sent by find key
key_help	**khlp**	**%1**	KEY_HELP, 0553, Sent by help key
key_home	**khome**	**kh**	KEY_HOME, 0406, Sent by home key
key_ic	**kich1**	**kI**	KEY_IC, 0513, Sent by ins-char/enter ins-mode key
key_il	**kil1**	**kA**	KEY_IL, 0511, Sent by insert-line key
key_left	**kcub1**	**kl**	KEY_LEFT, 0404, Sent by terminal left-arrow key
key_ll	**kll**	**kH**	KEY_LL, 0533, Sent by home-down key
key_mark	**kmrk**	**%2**	KEY_MARK, 0554, Sent by mark key
key_message	**kmsg**	**%3**	KEY_MESSAGE, 0555, Sent by message key
key_move	**kmov**	**%4**	KEY_MOVE, 0556, Sent by move key
key_next	**knxt**	**%5**	KEY_NEXT, 0557, Sent by next-object key
key_npage	**knp**	**kN**	KEY_NPAGE, 0522, Sent by next-page key
key_open	**kopn**	**%6**	KEY_OPEN, 0560, Sent by open key
key_options	**kopt**	**%7**	KEY_OPTIONS, 0561, Sent by options key
key_ppage	**kpp**	**kP**	KEY_PPAGE, 0523, Sent by previous-page key
key_previous	**kprv**	**%8**	KEY_PREVIOUS, 0562, Sent by previous-object key
key_print	**kprt**	**%9**	KEY_PRINT, 0532, Sent by print or copy key
key_redo	**krdo**	**%0**	KEY_REDO, 0563, Sent by redo key
key_reference	**kref**	**&1**	KEY_REFERENCE, 0564, Sent by ref(erence) key
key_refresh	**krfr**	**&2**	KEY_REFRESH, 0565, Sent by refresh key
key_replace	**krpl**	**&3**	KEY_REPLACE, 0566, Sent by replace key
key_restart	**krst**	**&4**	KEY_RESTART, 0567, Sent by restart key
key_resume	**kres**	**&5**	KEY_RESUME, 0570, Sent by resume key
key_right	**kcuf1**	**kr**	KEY_RIGHT, 0405, Sent by terminal right-arrow key
key_save	**ksav**	**&6**	KEY_SAVE, 0571, Sent by save key

key_sbeg	**kBEG**	**&9**	KEY_SBEG, 0572, Sent by shifted beginning key
key_scancel	**kCAN**	**&0**	KEY_SCANCEL, 0573, Sent by shifted cancel key
key_scommand	**kCMD**	***1**	KEY_SCOMMAND, 0574, Sent by shifted command key
key_scopy	**kCPY**	***2**	KEY_SCOPY, 0575, Sent by shifted copy key
key_screate	**kCRT**	***3**	KEY_SCREATE, 0576, Sent by shifted create key
key_sdc	**kDC**	***4**	KEY_SDC, 0577, Sent by shifted delete-char key
key_sdl	**kDL**	***5**	KEY_SDL, 0600, Sent by shifted delete-line key
key_select	**kslt**	***6**	KEY_SELECT, 0601, Sent by select key
key_send	**kEND**	***7**	KEY_SEND, 0602, Sent by shifted end key
key_seol	**kEOL**	***8**	KEY_SEOL, 0603, Sent by shifted clear-line key
key_sexit	**kEXT**	***9**	KEY_SEXIT, 0604, Sent by shifted exit key
key_sf	**kind**	**kF**	KEY_SF, 0520, Sent by scroll-forward/down key
key_sfind	**kFND**	***0**	KEY_SFIND, 0605, Sent by shifted find key
key_shelp	**kHLP**	**#1**	KEY_SHELP, 0606, Sent by shifted help key
key_shome	**kHOM**	**#2**	KEY_SHOME, 0607, Sent by shifted home key
key_sic	**kIC**	**#3**	KEY_SIC, 0610, Sent by shifted input key
key_sleft	**kLFT**	**#4**	KEY_SLEFT, 0611, Sent by shifted left-arrow key
key_smessage	**kMSG**	**%a**	KEY_SMESSAGE, 0612, Sent by shifted message key
key_smove	**kMOV**	**%b**	KEY_SMOVE, 0613, Sent by shifted move key
key_snext	**kNXT**	**%c**	KEY_SNEXT, 0614, Sent by shifted next key
key_soptions	**kOPT**	**%d**	KEY_SOPTIONS, 0615, Sent by shifted options key
key_sprevious	**kPRV**	**%e**	KEY_SPREVIOUS, 0616, Sent by shifted prev key
key_sprint	**kPRT**	**%f**	KEY_SPRINT, 0617, Sent by shifted print key
key_sr	**kri**	**kR**	KEY_SR, 0521, Sent by scroll-backward/up key
key_sredo	**kRDO**	**%g**	KEY_SREDO, 0620, Sent by shifted redo key
key_sreplace	**kRPL**	**%h**	KEY_SREPLACE, 0621, Sent by shifted replace key
key_sright	**kRIT**	**%i**	KEY_SRIGHT, 0622, Sent by shifted right-arrow key
key_srsume	**kRES**	**%j**	KEY_SRSUME, 0623, Sent by shifted resume key
key_ssave	**kSAV**	**!1**	KEY_SSAVE, 0624, Sent by shifted save key
key_ssuspend	**kSPD**	**!2**	KEY_SSUSPEND, 0625, Sent by shifted suspend key
key_stab	**khts**	**kT**	KEY_STAB, 0524, Sent by set-tab key
key_sundo	**kUND**	**!3**	KEY_SUNDO, 0626, Sent by shifted undo key
key_suspend	**kspd**	**&7**	KEY_SUSPEND, 0627, Sent by suspend key
key_undo	**kund**	**&8**	KEY_UNDO, 0630, Sent by undo key
key_up	**kcuu1**	**ku**	KEY_UP, 0403, Sent by terminal up-arrow key
keypad_local	**rmkx**	**ke**	Out of "keypad-transmit" mode
keypad_xmit	**smkx**	**ks**	Put terminal in "keypad-transmit" mode
lab_f0	**lf0**	**l0**	Labels on function key f0 if not f0
lab_f1	**lf1**	**l1**	Labels on function key f1 if not f1
lab_f2	**lf2**	**l2**	Labels on function key f2 if not f2
lab_f3	**lf3**	**l3**	Labels on function key f3 if not f3
lab_f4	**lf4**	**l4**	Labels on function key f4 if not f4
lab_f5	**lf5**	**l5**	Labels on function key f5 if not f5
lab_f6	**lf6**	**l6**	Labels on function key f6 if not f6
lab_f7	**lf7**	**l7**	Labels on function key f7 if not f7
lab_f8	**lf8**	**l8**	Labels on function key f8 if not f8
lab_f9	**lf9**	**l9**	Labels on function key f9 if not f9
lab_f10	**lf10**	**la**	Labels on function key f10 if not f10
label_off	**rmln**	**LF**	Turn off soft labels
label_on	**smln**	**LO**	Turn on soft labels
meta_off	**rmm**	**mo**	Turn off "meta mode"
meta_on	**smm**	**mm**	Turn on "meta mode" (8th bit)
newline	**nel**	**nw**	NEWLINE (behaves like **cr** followed by **lf**)
pad_char	**pad**	**pc**	Pad character (rather than null)
parm_dch	**dch**	**DC**	Delete #1 chars (G*)
parm_delete_line	**dl**	**DL**	Delete #1 lines (G*)
parm_down_cursor	**cud**	**DO**	Move cursor down #1 lines. (G*)
parm_ich	**ich**	**IC**	Insert #1 blank chars (G*)

parm_index	**indn**	**SF**	Scroll forward #1 lines. (G)
parm_insert_line	**il**	**AL**	Add #1 new blank lines (G*)
parm_left_cursor	**cub**	**LE**	Move cursor left #1 spaces (G)
parm_right_cursor	**cuf**	**RI**	Move cursor right #1 spaces. (G*)
parm_rindex	**rin**	**SR**	Scroll backward #1 lines. (G)
parm_up_cursor	**cuu**	**UP**	Move cursor up #1 lines. (G*)
pkey_key	**pfkey**	**pk**	Prog funct key #1 to type string #2
pkey_local	**pfloc**	**pl**	Prog funct key #1 to execute string #2
pkey_xmit	**pfx**	**px**	Prog funct key #1 to xmit string #2
plab_norm	**pln**	**pn**	Prog label #1 to show string #2
print_screen	**mc0**	**ps**	Print contents of the screen
prtr_non	**mc5p**	**pO**	Turn on the printer for #1 bytes
prtr_off	**mc4**	**pf**	Turn off the printer
prtr_on	**mc5**	**po**	Turn on the printer
repeat_char	**rep**	**rp**	Repeat char #1 #2 times (G*)
req_for_input	**rfi**	**RF**	Send next input char (for ptys)
reset_1string	**rs1**	**r1**	Reset terminal completely to sane modes
reset_2string	**rs2**	**r2**	Reset terminal completely to sane modes
reset_3string	**rs3**	**r3**	Reset terminal completely to sane modes
reset_file	**rf**	**rf**	Name of file containing reset string
restore_cursor	**rc**	**rc**	Restore cursor to position of last sc
row_address	**vpa**	**cv**	Vertical position absolute (G)
save_cursor	**sc**	**sc**	Save cursor position
scroll_forward	**ind**	**sf**	Scroll text up
scroll_reverse	**ri**	**sr**	Scroll text down
set_attributes	**sgr**	**sa**	Define the video attributes #1-#9 (G)
set_left_margin	**smgl**	**ML**	Set soft left margin
set_right_margin	**smgr**	**MR**	Set soft right margin
set_tab	**hts**	**st**	Set a tab stop in all rows, current column
set_window	**wind**	**wi**	Current window is lines #1-#2 cols #3-#4 (G)
tab	**ht**	**ta**	Move the cursor to the next 8 space hardware tab stop
to_status_line	**tsl**	**ts**	Go to status line, col #1 (G)
underline_char	**uc**	**uc**	Underscore one char and move past it
up_half_line	**hu**	**hu**	Half-line up (reverse 1/2 line-feed)
xoff_character	**xoffc**	**XF**	X-off character
xon_character	**xonc**	**XN**	X-on character

SAMPLE ENTRY

The following entry, which describes the Concept 100 terminal, is among the more complex entries in the **terminfo** file as of this writing.

```
concept100|c100|concept|c104|c100-4p|concept 100,
    am, db, eo, in, mir, ul, xenl, cols#80, lines#24, pb#9600, vt#8,
    bel=^G, blank=\EH, blink=\EC, clear=^L$<2*>, cnorm=\Ew, cr=^M$<9>,
    cub1=^H, cud1=^J, cuf1=\E=, cup=\Ea%p1%' '%+%c%p2%' '%+%c, cuu1=\E;,
    cvvis=\EW, dch1=\E^A$<16*>, dim=\EE, dl1=\E^B$<3*>,
    ed=\E^C$<16*>, el=\E^U$<16>, flash=\Ek$<20>\EK, ht=\t$<8>,
    il1=\E^R$<3*>, ind=^J, .ind=^J$<9>, ip=$<16*>,
    is2=\EU\Ef\E7\E5\E8\El\ENH\EK\E\0\Eo&\0\Eo\47\E,
    kbs=^h, kcub1=\E>, kcud1=\E<, kcuf1=\E=, kcuu1=\E;, kf1=\E5,
    kf2=\E6, kf3=\E7, khome=\E?, prot=\EI,
    rep=\Er%p1%c%p2%' '%+%c$<.2*>, rev=\ED,
    rmcup=\Ev\s\s\s\s$<6>\Ep\r\n, rmir=\E\0, rmkx=\Ex,
    rmso=\Ed\Ee, rmul=\Eg, rmul=\Eg, sgr0=\EN\0,
    smcup=\EU\Ev\s\s8p\Ep\r, smir=\E^P, smkx=\EX, smso=\EE\ED,
    smul=\EG,
```

Entries may continue onto multiple lines by placing white space at the beginning of each line except the first. Lines beginning with # are taken as comment lines. Capabilities in **terminfo** are of three types: boolean capabilities which indicate that the terminal has some particular feature, numeric capabilities giving the size of the terminal or particular features, and string capabilities, which give a sequence which can be used to perform particular terminal operations.

Types of Capabilities

All capabilities have names. For instance, the fact that the Concept has *automatic margins* (that is, an automatic RETURN and LINEFEED when the end of a line is reached) is indicated by the capability **am**. Hence the description of the Concept includes **am**. Numeric capabilities are followed by the character # and then the value. Thus **cols**, which indicates the number of columns the terminal has, gives the value **80** for the Concept. The value may be specified in decimal, octal or hexadecimal using normal C conventions.

Finally, string-valued capabilities, such as **el** (clear to end of line sequence) are given by the two- to five-character capname, an '=', and then a string ending at the next following comma. A delay in milliseconds may appear anywhere in such a capability, enclosed in **$<..>** brackets, as in '**el=\EK$<3>**', and padding characters are supplied by **tputs**() (see **curses**(3V)) to provide this delay. The delay can be either a number, for example, **20**, or a number followed by an * (for example, **3***), a / (for example, **5/**), or both (for example, **10*/**). A * indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. (In the case of insert character, the factor is still the number of lines affected. This is always one unless the terminal has **in** and the software uses it.) When a * is specified, it is sometimes useful to give a delay of the form **3.5** to specify a delay per unit to tenths of milliseconds. (Only one decimal place is allowed.) A / indicates that the padding is mandatory. Otherwise, if the terminal has **xon** defined, the padding information is advisory and will only be used for cost estimates or when the terminal is in raw mode. Mandatory padding will be transmitted regardless of the setting of **xon**.

A number of escape sequences are provided in the string-valued capabilities for easy encoding of characters there:

\E, \e	map to ESC
^X	maps to CTRL-*X* for any appropriate character *X*
\n	maps to NEWLINE
\l	maps to LINEFEED
\r	maps to RETURN
\t	maps to TAB
\b	maps to BACKSPACE
\f	maps to FORMFEED
\s	maps to SPACE
\0	maps to NUL

(**\0** will actually produce **\200**, which does not terminate a string but behaves as a null character on most terminals.) Finally, characters may be given as three octal digits after a backslash (for example, **\123**), and the characters ^ (caret), \ (backslash), : (colon), and , (comma) may be given as \^, \\, \:, and \, respectively.

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the second **ind** in the example above. Note: capabilities are defined in a left-to-right order and, therefore, a prior definition will override a later definition.

Preparing Descriptions

The most effective way to prepare a terminal description is by imitating the description of a similar terminal in **terminfo** and to build up a description gradually, using partial descriptions with some *curses*-based application to check that they are correct. Be aware that a very unusual terminal may expose deficiencies in the ability of the **terminfo** file to describe it or bugs in the application. To test a new terminal description, set the environment variable **TERMINFO** to a pathname of a directory containing the compiled description you are working on and programs will look there rather than in **/usr/share/lib/terminfo**. To get the padding for insert-line correct (if the terminal manufacturer did not document it) a severe test is to insert 16 lines into the middle of a full screen at 9600 baud. If the display is corrupted, more padding is usually needed. A similar test can be used for insert-character.

Basic Capabilities

The number of columns on each line for the terminal is given by the **cols** numeric capability. If the terminal has a screen, then the number of lines on the screen is given by the **lines** capability. If the terminal wraps around to the beginning of the next line when it reaches the right margin, then it should have the **am** capability. If the terminal can clear its screen, leaving the cursor in the home position, then this is given by the **clear** string capability. If the terminal overstrikes (rather than clearing a position when a character is struck over) then it should have the **os** capability. If the terminal is a printing terminal, with no soft copy unit, give it both **hc** and **os**. (**os** applies to storage scope terminals, such as Tektronix 4010 series, as well as hard-copy and APL terminals.) If there is a code to move the cursor to the left edge of the current row, give this as **cr**. (Normally this will be RETURN, CTRL-M.) If there is a code to produce an audible signal (bell, beep, etc) give this as **bel**. If the terminal uses the xon-xoff flow-control protocol, like most terminals, specify **xon**.

If there is a code to move the cursor one position to the left (such as backspace) that capability should be given as **cub1**. Similarly, codes to move to the right, up, and down should be given as **cuf1**, **cuu1**, and **cud1**. These local cursor motions should not alter the text they pass over; for example, you would not normally use **cuf1**=\s because the SPACE would erase the character moved over.

A very important point here is that the local cursor motions encoded in **terminfo** are undefined at the left and top edges of a screen terminal. Programs should never attempt to backspace around the left edge, unless **bw** is given, and should never attempt to go up locally off the top. In order to scroll text up, a program will go to the bottom left corner of the screen and send the **ind** (index) string.

To scroll text down, a program goes to the top left corner of the screen and sends the **ri** (reverse index) string. The strings **ind** and **ri** are undefined when not on their respective corners of the screen.

Parameterized versions of the scrolling sequences are **indn** and **rin** which have the same semantics as **ind** and **ri** except that they take one parameter, and scroll that many lines. They are also undefined except at the appropriate edge of the screen.

The **am** capability tells whether the cursor sticks at the right edge of the screen when text is output, but this does not necessarily apply to a **cuf1** from the last column. The only local motion which is defined from the left edge is if **bw** is given, then a **cub1** from the left edge will move to the right edge of the previous row. If **bw** is not given, the effect is undefined. This is useful for drawing a box around the edge of the screen, for example. If the terminal has switch selectable automatic margins, the **terminfo** file usually assumes that this is on; that is, **am**. If the terminal has a command which moves to the first column of the next line, that command can be given as **nel** (NEWLINE). It does not matter if the command clears the remainder of the current line, so if the terminal has no **cr** and **lf** it may still be possible to craft a working **nel** out of one or both of them.

These capabilities suffice to describe hardcopy and screen terminals. Thus the model 33 teletype is described as

33|tty33|tty|model 33 teletype,
bel=^G, cols#72, cr=^M, cud1=^J, hc, ind=^J, os,

while the Lear Siegler ADM–3 is described as

adm3|lsi adm3,
am, bel=^G, clear=^Z, cols#80, cr=^M, cub1=^H,
cud1=^J, ind=^J, lines#24,

Parameterized Strings

Cursor addressing and other strings requiring parameters in the terminal are described by a parameterized string capability, with **printf**(3V)-like escapes (**%x**) in it. For example, to address the cursor, the **cup** capability is given, using two parameters: the row and column to address to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory.) If the terminal has memory relative cursor addressing, that can be indicated by **mrcup**.

The parameter mechanism uses a stack and special % codes to manipulate it in the manner of a Reverse Polish Notation (postfix) calculator. Typically a sequence will push one of the parameters onto the stack and then print it in some format. Often more complex operations are necessary. Binary operations are in postfix form with the operands in the usual order. That is, to get x–5 one would use '**%gx%{5}%–**'.

The % encodings have the following meanings:

%%	outputs %
%[[:]*flags***] [***width***[.***precision***]] [doxXs]**	as in **printf**(3V), flags are [–+#] and SPACE
%c	print **pop**() gives **%c**
%p[1-9]	push i^{th} parm
%P[a-z]	set variable [a-z] to **pop**()
%g[a-z]	get variable [a-z] and push it
%'*c***'**	push char constant *c*
%{*nn***}**	push decimal constant *nn*
%l	push **strlen(pop())**
%+ %– %* %/ %m	arithmetic (**%m** is mod): **push(pop() op pop())**
%& %| %^	bit operations: **push(pop() op pop())**
%= %> %<	logical operations: **push(pop() op pop())**
%A %O	logical operations: and, or
%! %~	unary operations: **push(op pop())**
%i	(for ANSI terminals) add 1 to first parm, if one parm present, or first two parms, if more than one parm present
%?*expr* **%t***thenpart* **%e***elsepart***%;**	if-then-else, '**%e***elsepart*' is optional; else-if's are possible in Algol 68: **%? c_1 %t b_1 %e c_2 %t b_2 %e c_3 %t b_3 %e c_4 %t b_4 %e b_5%;** c_i are conditions, b_i are bodies.

If the '–' flag is used with '**%[doxXs]**', then a colon (:) must be placed between the '**%**' and the '–' to differentiate the flag from the binary '**%–**' operator, for example, '**%:–16.16s**'.

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent **\E&a12c03Y** padded for 6 milliseconds. Note: the order of the rows and columns is inverted here, and that the row and column are zero-padded as two digits. Thus its **cup** capability is:

cup=\E&a%p2%2.2dc%p1%2.2dY$<6>

The Micro-Term ACT-IV needs the current row and column sent preceded by a ^T, with the row and column simply encoded in binary, '**cup**=^T%p1%c%p2%c'. Terminals which use **%c** need to be able to backspace the cursor (**cub1**), and to move the cursor up one line on the screen (**cuu1**). This is necessary because it is not always safe to transmit **\n**, ^**D**, and **\r**, as the system may change or discard them. (The library routines dealing with **terminfo** set tty modes so that TAB characters are never expanded, so **\t** is safe to send. This turns out to be essential for the Ann Arbor 4080.)

A final example is the LSI ADM-3a, which uses row and column offset by a blank character, thus '**cup**=\E=%p1%'\s'%+%c%p2%'\s'%+%c'. After sending '\E=', this pushes the first parameter, pushes the ASCII value for a space (32), adds them (pushing the sum on the stack in place of the two previous values), and outputs that value as a character. Then the same is done for the second parameter. More complex arithmetic is possible using the stack.

Cursor Motions

If the terminal has a fast way to home the cursor (to very upper left corner of screen) then this can be given as **home**; similarly a fast way of getting to the lower left-hand corner can be given as **ll**; this may involve going up with **cuu1** from the home position, but a program should never do this itself (unless **ll** does) because it can make no assumption about the effect of moving up from the home position. Note: the home position is the same as addressing to (**0**,**0**): to the top left corner of the screen, not of memory. (Thus, the **\EH** sequence on Hewlett-Packard terminals cannot be used for **home** without losing some of the other features on the terminal.)

If the terminal has row or column absolute-cursor addressing, these can be given as single parameter capabilities **hpa** (horizontal position absolute) and **vpa** (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to **cup**. If there are parameterized local motions (for example, move *n* spaces to the right) these can be given as **cud**, **cub**, **cuf**, and **cuu** with a single parameter indicating how many spaces to move. These are primarily useful if the terminal does not have **cup**, such as the Tektronix 4025.

Area Clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as **el**. If the terminal can clear from the beginning of the line to the current position inclusive, leaving the cursor where it is, this should be given as **el1**. If the terminal can clear from the current position to the end of the display, then this should be given as **ed**. **ed** is only defined from the first column of a line. (Thus, it can be simulated by a request to delete a large number of lines, if a true **ed** is not available.)

Insert/Delete Line

If the terminal can open a new blank line before the line where the cursor is, this should be given as '**il1**'; this is done only from the first position of a line. The cursor must then appear on the newly blank line. If the terminal can delete the line which the cursor is on, then this should be given as '**dl1**'; this is done only from the first position on the line to be deleted. Versions of **il1** and **dl1** which take a single parameter and insert or delete that many lines can be given as **il** and **dl**.

If the terminal has a settable destructive scrolling region (like the VT100) the command to set this can be described with the **csr** capability, which takes two parameters: the top and bottom lines of the scrolling region. The cursor position is, alas, undefined after using this command. It is possible to get the effect of insert or delete line using this command — the **sc** and **rc** (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using **ri** or **ind** on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

To determine whether a terminal has destructive scrolling regions or non-destructive scrolling regions, create a scrolling region in the middle of the screen, place data on the bottom line of the scrolling region, move the cursor to the top line of the scrolling region, and do a reverse index (**ri**) followed by a delete line (**dl1**) or index (**ind**). If the data that was originally on the bottom line of the scrolling region was restored into the scrolling region by the **dl1** or **ind**, then the terminal has non-destructive scrolling regions. Otherwise, it has destructive scrolling regions. Do not specify **csr** if the terminal has non-destructive scrolling regions, unless **ind**, **ri**, **indn**, **rin**, **dl**, and **dl1** all simulate destructive scrolling.

If the terminal has the ability to define a window as part of memory, which all commands affect, it should be given as the parameterized string **wind**. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order.

If the terminal can retain display memory above, then the **da** capability should be given; if display memory can be retained below, then **db** should be given. These indicate that deleting a line or scrolling a full screen may bring non-blank lines up from below or that scrolling back with **ri** may bring down non-blank lines.

Insert/Delete Character

There are two basic kinds of intelligent terminals with respect to insert/delete character operations which can be described using **terminfo**. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept 100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks. You can determine the kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type '**abc def**' using local cursor motions (not SPACE characters) between the **abc** and the **def**. Then position the cursor before the **abc** and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the **abc** shifts over to the **def** which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal, and should give the capability **in**, which stands for "insert null". While these are two logically separate attributes (one line versus multiline insert mode, and special treatment of untyped blanks) we have seen no terminals whose insert mode cannot be described with the single attribute.

terminfo can describe both terminals which have an insert mode and terminals which send a simple sequence to open a blank position on the current line. Give as **smir** the sequence to get into insert mode. Give as **rmir** the sequence to leave insert mode. Now give as **ich1** any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode will not give **ich1**; terminals which send a sequence to open a screen position should give it here. (If your terminal has both, insert mode is usually preferable to **ich1**. Do not give both unless the terminal actually requires both to be used in combination.) If post-insert padding is needed, give this as a number of milliseconds padding in **ip** (a string option). Any other sequence which may need to be sent after an insert of a single character may also be given in **ip**. If your terminal needs both to be placed into an "insert mode" and a special code to precede each inserted character, then both **smir/rmir** and **ich1** can be given, and both will be used. The **ich** capability, with one parameter, *n*, will repeat the effects of **ich1** *n* times.

If padding is necessary between characters typed while not in insert mode, give this as a number of milliseconds padding in **rmp**.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (for example, if there is a TAB character after the insertion position). If your terminal allows motion while in insert mode you can give the capability **mir** to speed up inserting in this case. Omitting **mir** will affect only speed. Some terminals (notably Datamedia's) must not have **mir** because of the way their insert mode works.

Finally, you can specify **dch1** to delete a single character, **dch** with one parameter, *n*, to delete *n* characters, and delete mode by giving **smdc** and **rmdc** to enter and exit delete mode (any mode the terminal needs to be placed in for **dch1** to work).

A command to erase *n* characters (equivalent to outputting *n* blanks without moving the cursor) can be given as **ech** with one parameter.

Highlighting, Underlining, and Visible Bells

If your terminal has one or more kinds of display attributes, these can be represented in a number of different ways. You should choose one display form as *standout mode* (see **curses**(3V)), representing a good, high contrast, easy-on-the-eyes, format for highlighting error messages and other attention getters. (If you have a choice, reverse-video plus half-bright is good, or reverse-video alone; however, different users have

different preferences on different terminals.) The sequences to enter and exit standout mode are given as **smso** and **rmso**, respectively. If the code to change into or out of standout mode leaves one or even two blanks on the screen, as the TVI 912 and Teleray 1061 do, then **xmc** should be given to tell how many blanks are left.

Codes to begin underlining and end underlining can be given as **smul** and **rmul** respectively. If the terminal has a code to underline the current character and move the cursor one position to the right, such as the Micro-Term MIME, this can be given as **uc**.

Other capabilities to enter various highlighting modes include **blink** (blinking), **bold** (bold or extra-bright), **dim** (dim or half-bright), **invis** (blanking or invisible text), **prot** (protected), **rev** (reverse-video), **sgr0** (turn off all attribute modes), **smacs** (enter alternate-character-set mode), and **rmacs** (exit alternate-character-set mode). Turning on any of these modes singly may or may not turn off other modes. If a command is necessary before alternate character set mode is entered, give the sequence in **enacs** (enable alternate-character-set mode).

If there is a sequence to set arbitrary combinations of modes, this should be given as **sgr** (set attributes), taking nine parameters. Each parameter is either **0** or non-zero, as the corresponding attribute is on or off. The nine parameters are, in order: standout, underline, reverse, blink, dim, bold, blank, protect, alternate character set. Not all modes need be supported by **sgr**, only those for which corresponding separate attribute commands exist. (See the example at the end of this section.)

Terminals with the "magic cookie" glitch (**xmc**) deposit special "cookies" when they receive mode-setting sequences, which affect the display algorithm rather than having extra bits for each character. Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when they move to a new line or the cursor is addressed. Programs using standout mode should exit standout mode before moving the cursor or sending a newline, unless the **msgr** capability, asserting that it is safe to move in standout mode, is present.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), then this can be given as **flash**; it must not move the cursor. A good flash can be done by changing the screen into reverse video, pad for 200 ms, then return the screen to normal video.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to make, for example, a non-blinking underline into an easier to find block or blinking underline) give this sequence as **cvvis**. The boolean **chts** should also be given. If there is a way to make the cursor completely invisible, give that as **civis**. The capability **cnorm** should be given which undoes the effects of either of these modes.

If the terminal needs to be in a special mode when running a program that uses these capabilities, the codes to enter and exit this mode can be given as **smcup** and **rmcup**. This arises, for example, from terminals like the Concept with more than one page of memory. If the terminal has only memory relative cursor addressing and not screen relative cursor addressing, a one screen-sized window must be fixed into the terminal for cursor addressing to work properly. This is also used for the Tektronix 4025, where **smcup** sets the command character to be the one used by **terminfo**. If the **smcup** sequence will not restore the screen after an **rmcup** sequence is output (to the state prior to outputting **rmcup**), specify **nrrmc**.

If your terminal generates underlined characters by using the underline character (with no special codes needed) even though it does not otherwise overstrike characters, then you should give the capability **ul**. For terminals where a character overstriking another leaves both characters on the screen, give the capability **os**. If overstrikes are erasable with a blank, then this should be indicated by giving **eo**.

Example of highlighting: assume that the terminal under question needs the following escape sequences to turn on various modes.

tparm parameter	**attribute**	**escape sequence**
	none	\E[0m
p1	standout	\E[0;4;7m
p2	underline	\E[0;3m

p3	reverse	\E[0;4m
p4	blink	\E[0;5m
p5	dim	\E[0;7m
p6	bold	\E[0;3;4m
p7	invis	\E[0;8m
p8	protect	not available
p9	altcharset	^O (off) ^N(on)

Note: each escape sequence requires a **0** to turn off other modes before turning on its own mode. Also note that, as suggested above, *standout* is set up to be the combination of *reverse* and *dim*. Also, since this terminal has no *bold* mode, *bold* is set up as the combination of *reverse* and *underline*. In addition, to allow combinations, such as *underline+blink*, the sequence to use would be '**\E[0;3;5m**'. The terminal does not have *protect* mode, either, but that cannot be simulated in any way, so **p8** is ignored. The *altcharset* mode is different in that it is either **^O** or **^N** depending on whether it is off or on. If all modes were to be turned on, the sequence would be '**\E[0;3;4;5;7;8m^N**'.

Now look at when different sequences are output. For example, '**;3**' is output when either '**p2**' or '**p6**' is true, that is, if either *underline* or *bold* modes are turned on. Writing out the above sequences, along with their dependencies, gives the following:

sequence	**when to output**	**terminfo translation**
\E[0	always	\E[0
;3	if p2 or p6	%?%p2%p6%|%t;3%;
;4	if p1 or p3 or p6	%?%p1%p3%|%p6%|%t;4%;
;5	if p4	%?%p4%t;5%;
;7	if p1 or p5	%?%p1%p5%|%t;7%;
;8	if p7	%?%p7%t;8%;
m	always	m
^N or ^O	if p9 ^N, else ^O	%?%p9%t^N%e^O%;

Putting this all together into the **sgr** sequence gives:

sgr=\E[0%?%p2%p6%\|%t;3%;%?%p1%p3%\|%p6%\|%t;4%;%?%p5%t;5%;%?%p1%p5%
\|%t;7%;%?%p7%t;8%;m%?%p9%t^N%e^O%;,

Keypad

If the terminal has a keypad that transmits codes when the keys are pressed, this information can be given. Note: it is not possible to handle terminals where the keypad only works in local (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, give these codes as **smkx** and **rmkx**. Otherwise the keypad is assumed to always transmit.

The codes sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as **kcub1**, **kcuf1**, **kcuu1**, **kcud1**, and **khome** respectively. If there are function keys such as f0, f1, ..., f63, the codes they send can be given as **kf0**, **kf1**, ..., **kf63**. If the first 11 keys have labels other than the default f0 through f10, the labels can be given as **lf0**, **lf1**, ..., **lf10**. The codes transmitted by certain other special keys can be given: **kll** (home down), **kbs** (BACKSPACE), **ktbc** (clear all tab stops), **kctab** (clear the tab stop in this column), **kclr** (clear screen or erase key), **kdch1** (delete character), **kdl1** (delete line), **krmir** (exit insert mode), **kel** (clear to end of line), **ked** (clear to end of screen), **kich1** (insert character or enter insert mode), **kil1** (insert line), **knp** (next page), **kpp** (previous page), **kind** (scroll forward/down), **kri** (scroll backward/up), **khts** (set a tab stop in this column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, the other five keys can be given as **ka1**, **ka3**, **kb2**, **kc1**, and **kc3**. These keys are useful when the effects of a 3 by 3 directional pad are needed. Further keys are defined above in the capabilities list.

Strings to program function keys can be given as **pfkey**, **pfloc**, and **pfx**. A string to program their soft-screen labels can be given as **pln**. Each of these strings takes two parameters: the function key number to program (from 0 to 10) and the string to program it with. Function key numbers out of this range may

program undefined keys in a terminal-dependent manner. The difference between the capabilities is that **pfkey** causes pressing the given key to be the same as the user typing the given string; **pfloc** executes the string by the terminal in local mode; and **pfx** transmits the string to the computer. The capabilities **nlab**, **lw** and **lh** define how many soft labels there are and their width and height. If there are commands to turn the labels on and off, give them in **smln** and **rmln**. **smln** is normally output after one or more **pln** sequences to make sure that the change becomes visible.

Tabs and Initialization

If the terminal has hardware tab stops, the command to advance to the next tab stop can be given as **ht** (usually CTRL-I). A "backtab" command which moves leftward to the next tab stop can be given as **cbt**. By convention, if the teletype modes indicate that TAB characters are being expanded by the computer rather than being sent to the terminal, programs should not use **ht** or **cbt** even if they are present, since the user may not have the tab stops properly set. If the terminal has hardware tab stops which are initially set every *n* spaces when the terminal is powered up, the numeric parameter **it** is given, showing the number of spaces the tab stops are set to. This is normally used by '**tput init**' (see **tput**(1V)) to determine whether to set the mode for hardware TAB expansion and whether to set the tab stops. If the terminal has tab stops that can be saved in nonvolatile memory, the **terminfo** description can assume that they are properly set. If there are commands to set and clear tab stops, they can be given as **tbc** (clear all tab stops) and **hts** (set a tab stop in the current column of every row).

Other capabilities include: **is1**, **is2**, and **is3**, initialization strings for the terminal; **iprog**, the path name of a program to be run to initialize the terminal; and **if**, the name of a file containing long initialization strings. These strings are expected to set the terminal into modes consistent with the rest of the **terminfo** description. They must be sent to the terminal each time the user logs in and be output in the following order: run the program **iprog**; output **is1**; output **is2**; set the margins using **mgc**, **smgl** and **smgr**; set the tab stops using **tbc** and **hts**; print the file **if**; and finally output **is3**. This is usually done using the **init** option of **tput**(1V).

Most initialization is done with **is2**. Special terminal modes can be set up without duplicating strings by putting the common sequences in **is2** and special cases in **is1** and **is3**. Sequences that do a harder reset from a totally unknown state can be given as **rs1**, **rs2**, **rf**, and **rs3**, analogous to **is1**, **is2**, **is3**, and **if**. (The method using files, **if** and **rf**, is used for a few terminals, from **/usr/share/lib/tabset/***; however, the recommended method is to use the initialization and reset strings.) These strings are output by '**tput reset**', which is used when the terminal gets into a wedged state. Commands are normally placed in **rs1**, **rs2**, **rs3**, and **rf** only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set a terminal into 80-column mode would normally be part of **is2**, but on some terminals it causes an annoying glitch on the screen and is not normally needed since the terminal is usually already in 80-column mode.

If a more complex sequence is needed to set the tab stops than can be described by using **tbc** and **hts**, the sequence can be placed in **is2** or **if**.

If there are commands to set and clear margins, they can be given as **mgc** (clear all margins), **smgl** (set left margin), and **smgr** (set right margin).

Delays

Certain capabilities control padding in the terminal driver. These are primarily needed by hard-copy terminals, and are used by '**tput init**' to set tty modes appropriately. Delays embedded in the capabilities **cr**, **ind**, **cub1**, **ff**, and **tab** can be used to set the appropriate delay bits to be set in the tty driver. If **pb** (padding baud rate) is given, these values can be ignored at baud rates below the value of **pb**.

Status Lines

If the terminal has an extra "status line" that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, into which one can cursor address normally (such as the Heathkit H19's 25th line, or the 24th line of a VT100 which is set to a 23-line scrolling region), the capability **hs** should be given. Special strings that go to a given column of the status line and return from the status line can be given as **tsl** and **fsl**. (**fsl** must leave the cursor position in the same place it was before **tsl**. If necessary, the **sc** and **rc** strings can be included in **tsl** and **fsl** to get this effect.) The capability **tsl** takes one parameter, which is the column number of the status line the cursor is to be moved to.

If escape sequences and other special commands, such as TAB, work while in the status line, the flag **eslok** can be given. A string which turns off the status line (or otherwise erases its contents) should be given as **dsl**. If the terminal has commands to save and restore the position of the cursor, give them as **sc** and **rc**. The status line is normally assumed to be the same width as the rest of the screen, for example, **cols**. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded) the width, in columns, can be indicated with the numeric parameter **wsl**.

Line Graphics

If the terminal has a line drawing alternate character set, the mapping of glyph to character would be given in **acsc**. The definition of this string is based on the alternate character set used in the DEC VT100 terminal, extended slightly with some characters from the AT&T 4410v1 terminal.

glyph name	VT100+ character
arrow pointing right	+
arrow pointing left	,
arrow pointing down	.
solid square block	0
lantern symbol	I
arrow pointing up	–
diamond	‘
checker board (stipple)	a
degree symbol	f
plus/minus	g
board of squares	h
lower right corner	j
upper right corner	k
upper left corner	l
lower left corner	m
plus	n
scan line 1	o
horizontal line	q
scan line 9	s
left tee (├)	t
right tee (┤)	u
bottom tee (┴)	v
top tee (┬)	w
vertical line	x
bullet	~

The best way to describe a new terminal's line graphics set is to add a third column to the above table with the characters for the new terminal that produce the appropriate glyph when the terminal is in the alternate character set mode. For example,

glyph name	VT100+ char	new tty char
upper left corner	l	R
lower left corner	m	F
upper right corner	k	T
lower right corner	j	G
horizontal line	q	,
vertical line	x	.

Now write down the characters left to right, as in '**acsc**=lRmFkTjGq\,x.'.

Miscellaneous

If the terminal requires other than a null (zero) character as a pad, then this can be given as **pad**. Only the first character of the **pad** string is used. If the terminal does not have a pad character, specify **npc**.

If the terminal can move up or down half a line, this can be indicated with **hu** (half-line up) and **hd** (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as **ff** (usually CTRL-L).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters) this can be indicated with the parameterized string **rep**. The first parameter is the character to be repeated and the second is the number of times to repeat it. Thus, '**tparm(repeat_char, 'x', 10)**' is the same as '**xxxxxxxxxx**'.

If the terminal has a settable command character, such as the Tektronix 4025, this can be indicated with **cmdch**. A prototype command character is chosen which is used in all capabilities. This character is given in the **cmdch** capability to identify it. On some UNIX systems, when the environment variable CC is set to a single-character value, all occurrences of the prototype character are replaced with that character.

Terminal descriptions that do not represent a specific kind of known terminal, such as **switch**, **dialup**, **patch**, and **network**, should include the **gn** (generic) capability so that programs can complain that they do not know how to talk to the terminal. (This capability does not apply to **virtual** terminal descriptions for which the escape sequences are known.) If the terminal is one of those supported by the UNIX system virtual terminal protocol, the terminal number can be given as **vt**. A line-turn-around sequence to be transmitted before doing reads should be specified in **rfi**.

If the terminal uses xon/xoff handshaking for flow control, give **xon**. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters will not be transmitted. Sequences to turn on and off xon/xoff handshaking may be given in **smxon** and **rmxon**. If the characters used for handshaking are not ^S and ^Q (CTRL-S and CTRL-Q, respectively), they may be specified with **xonc** and **xoffc**.

If the terminal has a "meta key" which acts as a shift key, setting the 8th bit of any character transmitted, this fact can be indicated with **km**. Otherwise, software will assume that the 8th bit is parity and it will usually be cleared. If strings exist to turn this "meta mode" on and off, they can be given as **smm** and **rmm**.

If the terminal has more lines of memory than will fit on the screen at once, the number of lines of memory can be indicated with **lm**. A value of **lm**#0 indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

Media copy strings which control an auxiliary printer connected to the terminal can be given as **mc0**: print the contents of the screen, **mc4**: turn off the printer, and **mc5**: turn on the printer. When the printer is on, all text sent to the terminal will be sent to the printer. A variation, **mc5p**, takes one parameter, and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. If the text is not displayed on the terminal screen when the printer is on, specify **mc5i** (silent printer). All text, including **mc4**, is transparently passed to the printer while an **mc5p** is in effect.

Special Cases

The working model used by **terminfo** fits most terminals reasonably well. However, some terminals do not completely match that model, requiring special support by **terminfo**. These are not meant to be construed as deficiencies in the terminals; they are just differences between the working model and the actual hardware. They may be unusual devices or, for some reason, do not have all the features of the **terminfo** model implemented.

Terminals which can not display tilde (~) characters, such as certain Hazeltine terminals, should indicate **hz**.

Terminals which ignore a LINEFEED immediately after an **am** wrap, such as the Concept 100, should indicate **xenl**. Those terminals whose cursor remains on the right-most column until another character has been received, rather than wrapping immediately upon receiving the right-most character, such as the VT100, should also indicate **xenl**.

If **el** is required to get rid of standout (instead of writing normal text on top of it), **xhp** should be given.

Those Teleray terminals whose tabs turn all characters moved over to blanks, should indicate **xt** (destructive TAB characters). This capability is also taken to mean that it is not possible to position the cursor on top of a "magic cookie" therefore, to erase standout mode, it is instead necessary to use delete and insert line.

Those Beehive Superbee terminals which do not transmit the escape or CTRL-C characters, should specify **xsb**, indicating that the f1 key is to be used for escape and the f2 key for CTRL-C.

Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability **use** can be given with the name of the similar terminal. The capabilities given before **use** override those in the terminal type invoked by **use**. A capability can be canceled by placing *xx*@ to the left of the capability definition, where *xx* is the capability. For example, the entry

att4424-2|Teletype 4424 in display function group ii,
rev@, sgr@, smul@, use=att4424,

defines an AT&T 4424 terminal that does not have the **rev**, **sgr**, and **smul** capabilities, and hence cannot do highlighting. This is useful for different modes for a terminal, or for different user preferences. More than one **use** capability may be given.

FILES

/usr/share/lib/terminfo/?/*
compiled terminal description database

/usr/share/lib/tabset/* tab stop settings for some terminals, in a format appropriate to be output to the terminal (escape sequences that set margins and tab stops)

SEE ALSO

tput(1V), **curses**(3V), **printf**(3V), **term**(5V), **captoinfo**(8V), **infocmp**(8V), **tic**(8V)

WARNING

As described in the **Tabs and Initialization** section above, a terminal's initialization strings, **is1**, **is2**, and **is3**, if defined, must be output before a **curses**(3V) program is run. An available mechanism for outputting such strings is **tput init** (see **tput**(1V)).

Tampering with entries in **/usr/share/lib/terminfo/?/*** (for example, changing or removing an entry) can affect programs that expect the entry to be present and correct. In particular, removing the description for the "dumb" terminal will cause unexpected problems.

NAME

toc – table of contents of optional clusters in Application SunOS and Developer's Toolkit

SYNOPSIS

/usr/lib/load/toc

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **toc** file contains information specifying the organization of the optional clusters in Application SunOS and Developer's Toolkit on the Sun386i distribution media. For each cluster, a single line should be present with the following information:

cluster name
set containing the cluster (Application SunOS or Developer's Toolkit)
size of the cluster (in kilobytes)
diskette volume of the cluster in the set (for loading from 3.5" diskette)
tape and file number of the cluster (for loading from 1/4" tape)

Items are separated by a ':'.

Cluster names can contain any printable character other than a ':', space, tab, or newline character. The set containing the cluster is specified by an 'A' for Application SunOS or 'D' for Developer's Toolkit. The diskette volume is the number of the diskette within the diskette set on which the cluster begins. The tape and file number specifies the tape and file position of the cluster on the tape.

EXAMPLE

The following is an example to the **toc** file.

accounting:A:55:14:1@12
advanced_admin:A:628:14:1@4
audit:A:144:14:1@8
comm:A:312:13:1@9
disk_quotas:A:56:14:1@11
doc_prep:A:790:13:1@10
extended_commands:A:276:13:1@5
games:A:2351:19:1@17
mail_plus:A:135:14:1@7
man_pages:A:5586:16:1@14
name_server:A:339:14:1@13
networking_plus:A:610:13:1@6
old:A:131:14:1@16
plot:A:227:14:1@14
spellcheck:A:455:13:1@2
sysV_commands:A:2505:14:1@3
base_devel:D:5389:1:2@2
plot_devel:D:247:5:2@3
sccs:D:328:5:2@4
sunview_devel:D:1768:5:2@5
sysV_devel:D:4287:3:2@6
proflibs:D:4755:4:2@7
config:D:3065:6:2@8

The first line specifies that the **accounting** cluster is part of Application SunOS and requires 55 kilobytes of disk storage. In the diskette distribution, it begins on diskette 14 of Application SunOS optional clusters. In the tape distribution, it can be found on file 12 of tape 1. The last line specifies that the *config* cluster is part of Developer's Toolkit and requires 3065 kilobytes of disk storage. In the diskette distribution, it begin on diskette 6 of Developer's Toolkit. In the tape distribution, it can be found on file 8 of tape 2.

FILES

/usr/lib/load/toc

SEE ALSO

cluster(1) **load**(1) **unload**(1)

NAME

translate – input and output files for system message translation

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

These files are used by **syslogd**(8) to translate systems messages. The input file is used to map system messages (in **printf**(3V) format strings) to numbers. This number is then used to locate a new string in the output file.

An initial part of each line in the input file may specify that the message should be suppressed. Recognized suppression specifications are:

(NONE) Suppress the message always.
(n) Allow only one message every n seconds. ((10) for example).
() Do not suppress the message. This can be used in a message that begins with a '('.

Note that the message suppression specification is optional. If not present, the message is not suppressed.

Each line in the output file translates the numbers from the input file into the desired error messages, and also specifies the format to be used to output each message. The order of parameters passed from the input message can be changed, by replacing the % of a format phrase with a *%num*$ where *num* is a digit string. For example, if *num* is 2, the second parameter on the input file line will be used. The value of *num* can be from 1 to the number of parameters in the input message.

If a string is translated to a number that is not found in the output file, the message is suppressed.

EXAMPLES

An example input file:

```
$quote "
1       "(NONE)(1) logopen test code: %s\n"
2       "(10)(2) logopen test code: %s\n"
3       "()(3) logopen test code: %s\n"
4       "()(4) logopen test code: %s\n"
5       "(10)(5) logopen testcode: %s * 100\n"
6       "(10)(6) logopen testcode: %s * 100\n"
7       "(10)(7) logopen testcode: %s * 100\n"
8       "(10)%s: %s\n"
9       "(10)\n%s: write failed, file system is full\n"
10      "(10)NFS server %s not responding still trying\n"
11      "(10)NFS %s failed for server %s: %s\n"
12      "(10)NFS server %s ok\n"
13      "(NONE)\n%s: write failed, file system is full\n"
14      "(10)NFS server %s not responding still trying\n"
15      "(100)NFS %s failed for server %s: %s\n"
```

An example output file:

```
$quote "
1       "TRANSLATION:(1) logopen test code: %s\n"
2       "TRANSLATION: (2) logopen test code: %s IS REALLY\n"
3       "TRANSLATION: (3) logopen test code: %s\n"
4       "TRANSLATION: (4) logopen test code: %s\n"
5       "TRANSLATION: (5) logopen testcode: %s * 100\n"
6       "TRANSLATION: (6) logopen testcode: %s * 100\n"
7       "TRANSLATION: (7) logopen testcode: %s * 100\n"
8       "TRANSLATION: %s: %s\n"
9       "TRANSLATION: \n%s: write failed, file system is full\n"
10      "TRANSLATION: NFS server %s not responding still trying\n"
11      "TRANSLATION: NFS %s failed for server %s: %s\n"
12      "TRANSLATION: NFS server %s ok\n"
13      "Out of disk on file system %s\n"
14      "Network file server %s not ok. Check your cable\n"
15      "Network file server %2$s down (%1$s, %3$s)\n"
```

SEE ALSO

syslogd(8)

NAME

ttytab, ttys – terminal initialization data

DESCRIPTION

The **/etc/ttytab** file contains information that is used by various routines to initialize and control the use of terminal special files. This information is read with the **getttyent**(3) library routines. There is one line in **/etc/ttytab** file per special file.

The **/etc/ttys** file should not be edited; it is derived from **/etc/ttytab** by **init(8)** at boot time, and is only included for backward compatibility with programs that may still require it.

Fields are separated by TAB and/or SPACE characters. Some fields may contain more than one word and should be enclosed in double quotes. Blank lines and comments can appear anywhere in the file; comments are delimited by '**#**' and NEWLINE. Unspecified fields default to NULL. The first field is the terminal's entry in the device directory, **/dev**. The second field of the file is the command to execute for the line, typically **getty**(8), which performs such tasks as baud-rate recognition, reading the login name, and calling **login**(1). It can be, however, any desired command, for example the start up for a window system terminal emulator or some other daemon process, and can contain multiple words if quoted. The third field is the type of terminal normally connected to that tty line, as found in the **termcap**(5) data base file. The remaining fields set flags in the **ty_status** entry (see **getttyent**(3)) or specify a window system process that **init**(8) will maintain for the terminal line.

As flag values, the strings **on** and **off** specify whether **init** should execute the command given in the second field, while **secure** in addition to **on** allows "root" to login on this line. If the console is not marked "secure," the system prompts for the root password before coming up in single-user mode. **local** in addition to **on** indicates that the line is a "local" line; the modem control signals for this line, such as Carrier Detect, will be ignored. These flag fields should not be quoted. The string **window=** is followed by a quoted command string which **init** will execute before starting **getty**.

The flag **local** applies to terminals, and enables the software carrier mode in the kernel; the kernel ignores the state of carrier detect when opening the serial port. Alternately, if this field is set to any value other than **local**, this flag disables the software carrier mode in the kernel, so the state of the carrier detect is not ignored. This usually applies to modems. See **termio**(4).

If the line ends in a comment, the comment is included in the **ty_comment** field of the **ttyent** structure.

After changing the **/etc/ttytab** file, you must notify **init**(8) before those changes will take effect. To do this, use:

kill –1 1

EXAMPLES

Below is a sample **/etc/ttytab** file:

```
console "/usr/etc/getty std.1200"  vt100      on secure
ttyd0   "/usr/etc/getty d1200"     dialup     on          # 555-1234
ttyh0   "/usr/etc/getty std.9600"  hp2621-nl  on          # 254MC
ttyh1   "/usr/etc/getty std.9600"  plugboard  on          # John's office
ttyp0   none                       network
ttyp1   none                       network    off
ttyv0   "/usr/new/xterm –L :0"     vs100      on window="/usr/new/Xvs100 0"
console "/usr/etc/getty –n –s   std.9600" sun      on    secure
console "/usr/etc/getty –n –s –l std.9600" sun      on    secure
```

The first line permits "root" login on the console at 1200 baud, and indicates that the console is physically secure for single-user operation. The second line allows dialup at 1200 baud without "root" login, and the third and fourth lines allow login at 9600 baud with terminal types of **hp2621-nl** and **plugboard**, respectively. The fifth and sixth lines are examples of network pseudo-ttys, **ttyp0** and **ttyp1** for which **getty** should not be enabled. The seventh line shows a terminal emulator and window-system startup entry. The last two lines instruct **getty,** using the **–n** argument, to run the **logintool**(8) graphic login interface, and the **–s** argument instructing **logintool** to start **screenblank**(1) with a plain black screen. The **–l** (lower case L) argument instructs **logintool** to start **lockscreen**(1). **lockscreen** starts after 30 minutes; there is no way to change this interval.

FILES

/dev
/etc/ttys
/etc/ttytab

SEE ALSO

login(1), **ioctl**(2), **getttyent**(3), **termio**(4), **gettytab**(5), **termcap**(5), **getty**(8), **init**(8), **logintool**(8), **ttysoftcar**(8)

NAME

types – primitive system data types

SYNOPSIS

#include <sys/types.h>

DESCRIPTION

The data types defined in the include file are used in the system code; some data of these types are accessible to user code:

```
/*
 * Copyright (c) 1982, 1986 Regents of the University of California.
 * All rights reserved. The Berkeley software License Agreement
 * specifies the terms and conditions for redistribution.
 */

#ifndef _TYPES_
#define _TYPES_

/*
 * Basic system types.
 */

#include <sys/sysmacros.h>

typedef unsigned char   u_char;
typedef unsigned short  u_short;
typedef unsigned int    u_int;
typedef unsigned long   u_long;
typedef unsigned short  ushort;/* System V compatibility */
typedef unsigned int    uint;/* System V compatibility */

#ifdef vax
typedef struct      _physadr { int r[1]; } *physadr;
typedef struct      label_t{
        int         val[14];
} label_t;
#endif
#ifdef mc68000
typedef struct      _physadr { short r[1]; } *physadr;
typedef struct      label_t{
        int         val[13];
} label_t;
#endif
#ifdef sparc
typedef struct  _physadr { int r[1]; } *physadr;
typedef struct label_t {
        int    val[2];
} label_t;
#endif
#ifdef i386
typedef struct      _physadr { short r[1]; } *physadr;
typedef struct      label_t {
        int         val[8];
} label_t;
```

```
#endif
typedef struct    _quad { long val[2]; } quad;
typedef long      daddr_t;
typedef char *    caddr_t;
typedef u_long    ino_t;
typedef long      swblk_t;
typedef int       size_t;
typedef long      time_t;
typedef short     dev_t;
typedef long      off_t;
typedef u_short   uid_t;
typedef u_short   gid_t;
typedef long      key_t;

#define NBBY    8       /* number of bits in a byte */
/*
 * Select uses bit masks of file descriptors in longs.
 * These macros manipulate such bit fields (the filesystem macros use chars).
 * FD_SETSIZE may be defined by the user, but the default here
 * should be >= NOFILE (param.h).
 */
#ifndef FD_SETSIZE
#define FD_SETSIZE    256
#endif

typedef long      fd_mask;
#define NFDBITS         (sizeof(fd_mask) * NBBY)/* bits per mask */
#ifndef howmany
#ifdef sun386
#define howmany(x, y)   ((((u_int)(x))+(((u_int)(y))-1))/((u_int)(y)))
#else
#define howmany(x, y)   (((x)+((y)-1))/(y))
#endif
#endif

typedef struct fd_set {
        fd_mask  fds_bits[howmany(FD_SETSIZE, NFDBITS)];
} fd_set;

typedef char *    addr_t;

#define FD_SET(n, p)    ((p)->fds_bits[(n)/NFDBITS] |= (1 << ((n) % NFDBITS)))
#define FD_CLR(n, p)    ((p)->fds_bits[(n)/NFDBITS] &= ~(1 << ((n) % NFDBITS)))
#define FD_ISSET(n, p) ((p)->fds_bits[(n)/NFDBITS] & (1 << ((n) % NFDBITS)))
#define FD_ZERO(p)      bzero((char *)(p), sizeof(*(p)))

#ifdef sparc
/*
 * routines that call setjmp have strange control flow graphs,
 * since a call to a routine that calls resume/longjmp will eventually
 * return at the setjmp site, not the original call site.  This
 * utterly wrecks control flow analysis.
 */
```

```
extern int setjmp();
#pragma unknown_control_flow(setjmp)
#endif sparc

#endif  _TYPES_
```

The form *daddr_t* is used for disk addresses, see **fs**(5). Times are encoded in seconds since 00:00:00 GMT, January 1, 1970. The major and minor parts of a device code specify kind and unit number of a device and are installation-dependent. Offsets are measured in bytes from the beginning of a file. The *label_t* variables are used to save the processor state while another process is running.

SEE ALSO

adb(1), **lseek**(2V), **time**(3V), **fs**(5)

NAME

tzfile – time zone information

SYNOPSIS

#include <tzfile.h>

DESCRIPTION

The time zone information files used by **tzset** (see **ctime**(3V)) begin with bytes reserved for future use, followed by three four-byte values of type **long**, written in a "standard" byte order (the high-order byte of the value is written first). These values are, in order:

tzh_timecnt	The number of "transition times" for which data is stored in the file.
tzh_typecnt	The number of "local time types" for which data is stored in the file (must not be zero).
tzh_charcnt	The number of characters of "time zone abbreviation strings" stored in the file.

The above header is followed by *tzh_timecnt* four-byte values of type **long**, sorted in ascending order. These values are written in "standard" byte order. Each is used as a transition time (as returned by **gettimeofday**(2)) at which the rules for computing local time change. Next come *tzh_timecnt* one-byte values of type **unsigned char**; each one tells which of the different types of "local time" types described in the file is associated with the same-indexed transition time. These values serve as indices into an array of *ttinfo* structures that appears next in the file; these structures are defined as follows:

```
struct ttinfo {
        long            tt_gmtoff;
        int             tt_isdst;
        unsigned int    tt_abbrind;
};
```

Each structure is written as a four-byte value for *tt_gmtoff* of type **long**, in a standard byte order, followed by a one-byte value for *tt_isdst* and a one-byte value for *tt_abbrind*. In each structure, *tt_gmtoff* gives the number of seconds to be added to GMT, *tt_isdst* tells whether *tm_isdst* should be set by **localtime** (see **ctime**(3V)) and *tt_abbrind* serves as an index into the array of time zone abbreviation characters that follow the *ttinfo* structure(s) in the file.

localtime uses the first standard-time *ttinfo* structure in the file (or simply the first *ttinfo* structure in the absence of a standard-time structure) if either *tzh_timecnt* is zero or the time argument is less than the first transition time recorded in the file.

SEE ALSO

gettimeofday(2), **ctime**(3V)

NAME

ugid_alloc.range – range of user IDs and group IDs to allocate

SYNOPSIS

/etc/ugid_alloc.range

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **/etc/ugid_alloc.range** file, if it exists on the Network Information Service (NIS) master of the **passwd.byuid** map (or the **group.bygid** map for group IDs), specifies the user IDs and group IDs that can be allocated for the local NIS domain by the **uid_allocd**(8C) daemons. If the file does not exist, user IDs or group IDs may be allocated beginning at 100 and ending at 60,000; no user IDs or group IDs are allocated out of that range in any case. If the local NIS domain is not listed in this file, no user IDs or group IDs will be allocated. Otherwise, this file specifies ranges of user IDs or group IDs that may be allocated. The different NIS domains on a network can use identical copies of this file.

If a network has multiple NIS domains, each one will typically use ranges for its user IDs and group IDs that do not overlap with the other NIS domains, guaranteeing that user IDs and group IDs are unique throughout the network. Without guarantees of user ID and group ID uniqueness, network tools and services which rely on that uniqueness for security or authentication will not work as intended. Such services include NFS, except for the "Secure NFS," which has other solutions for security and authentication. Note: the required uniqueness could be guaranteed by mechanisms other than automatic allocation within manually configured ranges. For example, some sites can use a function of their employee numbers during manual user ID allocation, and coordinate group ID assignment verbally.

This file can contain blank lines. Comments begin with a '#' character and extend to the end of the current line. The first token on the line is an NIS domain name. It is separated from the second token by white space (SPACE or TAB characters). The second token is either *user* or *group*, indicating that the line specifies user ID or group ID ranges, respectively. The third token is a comma-separated list of user or group ID ranges in that domain. These ranges take two forms: a single number specifies just that ID, and two numbers separated by a dash specify all IDs starting at the first number and ending with the second.

For example, the following file would direct that the manufacturing department at a particular company use user IDs from 700 to 999 or 1200 to 1499. Accounts created by tools in the NIS domain for manufacturing would use a user ID in those ranges, and those user accounts could safely be added to one of the other NIS domains if desired (by manually transferring NIS map data between the domains). Group IDs are allocated only within the administration domain.

```
# Three departments share our site's network, and each has its
# own Ethernet and master server connected with IP routers.
# This file sets the user ID ranges assigned to each department.
# Groups are defined by the administration group only.
YP.admin.company.com              user    500-699
YP.manufacturing.company.com      user    700-999
YP.engineering.company.com        user    100-499,1000-1199
YP.manufacturing.company.com      user    1200-1499
YP.admin.company.com              group   100-60000
```

SEE ALSO

passwd(5), **group**(5), **uid_allocd**(8C)

BUGS

There is a limit of forty ranges for each domain; more ranges are silently ignored.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

updaters – configuration file for NIS updating

SYNOPSIS

/var/yp/updaters

DESCRIPTION

The file **/var/yp/updaters** is a makefile (see **make**(1)) which is used for updating the Network Information Service (NIS) databases. Databases can only be updated in a secure network, that is, one that has a **publickey**(5) database. Each entry in the file is a make target for a particular NIS database. For example, if there is an NIS database named **passwd.byname** that can be updated, there should be a **make** target named **passwd.byname** in the **updaters** file with the command to update the file.

The information necessary to make the update is passed to the update command through standard input. The information passed is described below (all items are followed by a NEWLINE, except for 4 and 6)

- Network name of client wishing to make the update (a string)
- Kind of update (an integer)
- Number of bytes in key (an integer)
- Actual bytes of key
- Number of bytes in data (an integer)
- Actual bytes of data

After getting this information through standard input, the command to update the particular database should decide whether the user is allowed to make the change. If not, it should exit with the status YPERR_ACCESS. If the user is allowed to make the change, the command should make the change and exit with a status of zero. If there are any errors that may prevent the updater from making the change, it should exit with the status that matches a valid NIS error code described in **<rpcsvc/ypclnt.h>**.

FILES

/var/yp/updaters

SEE ALSO

make(1), **ypupdate**(3N), **publickey**(5), **ypupdated**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

utmp, wtmp, lastlog – login records

SYNOPSIS

#include <utmp.h>
#include <lastlog.h>

DESCRIPTION

utmp file

The **utmp** file records information about who is currently using the system. The file is a sequence of **utmp** structure entries. That structure is defined in **<utmp.h>**, and contains the following members:

ut_line	Character array containing the name of the terminal on which the user logged in.
ut_name	Character array containing the name of the user who logged in.
ut_host	Character array containing the name of the host from which the user remotely logged in, if they logged in from another host; otherwise, a null string.
ut_time	**long** containing the time at which the user logged in, in seconds since 00:00 GMT, January 1, 1970.

Whenever a user logs in, **login**(1) fills in the entry in **/etc/utmp** for the terminal on which the user logged in. When they log out, **init**(8) clears that entry by setting **ut_name** and **ut_host** to null strings and **ut_time** to the time at which the user logged out.

Some window systems will make entries in **utmp** for terminal emulation windows running shells, so that library routines such as **getlogin** will work correctly in that window. These entries do not directly represent logged-in users; they are associated with a user who has already logged into the system on another terminal. These entries generally have a **ut_line** field that refers to a pseudo-terminal, and a **ut_host** field that is a null string. The macro **nonuser**, defined in **<utmp.h>**, takes a pointer to a **utmp** structure as an argument and, if the entry has a **ut_line** field that refers to a pseudo-terminal, and a **ut_host** field that is a null string, will return 1; otherwise, it will return 0. This can be used by programs that print information about logged-in users if they should not list entries made for logged-in users' additional windows.

wtmp file

The **wtmp** file records all logins and logouts. It also consists of a sequence of **utmp** entries.

Whenever a user logs in, **login** appends a record identical to the record it placed in **utmp** to the end of **/var/adm/wtmp.** Whenever a user logs out, **init** appends a record with **ut_line** equal to the terminal that the user was logged in on, **ut_name** and **ut_host** null, and **ut_time** equal to the time at which the user logged out.

When the system is shut down, **init** appends a record with a **ut_line** of ~, a **ut_name** of **shutdown**, a null **ut_host**, and a **ut_time** equal to the time at which the shutdown occurred. When the system is rebooted, **init** appends a record with a **ut_line** of ~, a **ut_name** of **reboot**, a null **ut_host**, and a **ut_time** equal to the time at which **init** wrote the record.

When the **date** command is used to change the system-maintained time, **date** appends a record with a **ut_line** of |, **ut_name** and **ut_host** null, and **ut_time** equal to the system time before the change, and then appends a record with a **ut_line** of {, **ut_name** and **ut_host** null, and **ut_time** equal to the system time after the change.

None of the programs that maintain **wtmp** create the file, so that if record-keeping is to be enabled, it must be created by hand as a zero-length file, and if it is removed, record-keeping is turned off. It is summarized by **ac**(8).

As **wtmp** is appended to whenever a user logs in or out, it should be truncated periodically so that it does not consume all the disk space on its file system.

lastlog file

The **lastlog** file records the most recent login-date for every user logged in. The file is a sequence of **lastlog** structure entries. That structure is defined in **<lastlog.h>**, and contains the following members:

ll_time **long** containing the time at which the user logged in, in seconds since 00:00 GMT, January 1, 1970.

ll_line Character array containing the name of the terminal on which the user logged in.

ll_host Character array containing the name of the host from which the user remotely logged in, if they logged in from another host; otherwise, a null string.

When reporting (and updating) the most recent login date, **login** performs an **lseek**(2V) to a byte-offset in **/var/adm/lastlog** corresponding to the userid. Because the count of userids may be high, whereas the number actual users may be small within a network environment, the bulk of this file may never be allocated by the file system even though an offset may appear to be quite large. Although **ls**(1V) may show it to be large, chances are that this file need not be truncated. **du**(1V) will report the correct (smaller) amount of space actually allocated to it.

SYSTEM V DESCRIPTION

For XPG2 conformance, the XPG2 private **utmp** structure is preserved for use by compliant applications that specifically use the **utmp structure.** The structure is defined in **/usr/xpg2include/utmp.h**. Note: this structure definition was removed in XPG3, and will be removed in a future SunOS release. Applications using the XPG2 **utmp** structure must do so on an application private basis.

FILES

/etc/utmp
/var/adm/wtmp
/var/adm/lastlog

SEE ALSO

login(1), **who**(1), **ac**(8), **init**(8)

NAME

uuencode – format of an encoded uuencode file

DESCRIPTION

Files output by **uuencode**(1C) consist of a header line, followed by a number of body lines, and a trailer line. **uudecode** (see **uuencode**(1C)) will ignore any lines preceding the header or following the trailer. Lines preceding a header must not, of course, look like a header.

The header line is distinguished by having the first 6 characters '**begin** '. The word **begin** is followed by a mode (in octal), and a string which names the remote file. Spaces separate the three items in the header line.

The body consists of a number of lines, each at most 62 characters long (including the trailing NEWLINE). These consist of a character count, followed by encoded characters, followed by a NEWLINE. The character count is a single printing character, and represents an integer, the number of bytes the rest of the line represents. Such integers are always in the range from 0 to 63 and can be determined by subtracting the character space (octal 40) from the character.

Groups of 3 bytes are stored in 4 characters, 6 bits per character. All are offset by a SPACE to make the characters printing. The last line may be shorter than the normal 45 bytes. If the size is not a multiple of 3, this fact can be determined by the value of the count on the last line. Extra garbage will be included to make the character count a multiple of 4. The body is terminated by a line with a count of zero. This line consists of one ASCII SPACE.

The trailer line consists of **end** on a line by itself.

SEE ALSO

mail(1), **uucp**(1C), **uuencode**(1C), **uusend**(1C)

NAME

vfont – font formats

SYNOPSIS

#include <vfont.h>

DESCRIPTION

The fonts used by the window system and printer/plotters have the following format. Each font is in a file, which contains a header, an array of character description structures, and an array of bytes containing the bit maps for the characters. The header has the following format:

```
struct header {
        short           magic;          /* Magic number VFONT_MAGIC */
        unsigned short  size;           /* Total # bytes of bitmaps */
        short           maxx;           /* Maximum horizontal glyph size */
        short           maxy;           /* Maximum vertical glyph size */
        short           xtend;          /* (unused) */
};
#define VFONT_MAGIC                     0436
```

maxx and *maxy* are intended to be the maximum horizontal and vertical size of any glyph in the font, in raster lines. (A glyph is just a printed representation of a character, in a particular size and font.) The **size** is the total size of the bit maps for the characters in bytes. The *xtend* field is not currently used.

After the header is an array of NUM_DISPATCH structures, one for each of the possible characters in the font. Each element of the array has the form:

```
struct dispatch {
        unsigned short  addr;           /* &(glyph) - &(start of bitmaps) */
        short           nbytes;         /* # bytes of glyphs (0 if no glyph) */
        char            up, down, left, right;  /* Widths from baseline point */
        short           width;          /* Logical width, used by troff */
};
#define NUM_DISPATCH                    256
```

The *nbytes* field is nonzero for characters which actually exist. For such characters, the *addr* field is an offset into the bit maps to where the character's bit map begins. The *up* , *down*, *left*, and *right* fields are offsets from the base point of the glyph to the edges of the rectangle which the bit map represents. (The imaginary "base point" is a point which is vertically on the "base line" of the glyph (the bottom line of a glyph which does not have a descender) and horizontally near the left edge of the glyph; often 3 or so pixels past the left edge.) The bit map contains *up+down* rows of data for the character, each of which has *left+right* columns (bits). Each row is rounded up to a number of bytes. The *width* field represents the logical width of the glyph in bits, and shows the horizontal displacement to the base point of the next glyph.

FILES

/usr/lib/vfont/*
/usr/lib/fonts/fixedwidthfonts/*

SEE ALSO

troff(1), **vfontinfo**(1), **vswap**(1)

BUGS

A machine-independent font format should be defined. The **shorts** in the above structures contain different bit patterns depending whether the font file is for use on a VAX or a Sun. The **vswap** program must be used to convert one to the other.

NAME

vgrindefs – vgrind's language definition data base

SYNOPSIS

/usr/lib/vgrindefs

DESCRIPTION

vgrindefs contains all language definitions for **vgrind**(1). The data base is very similar to **termcap**(5). Capabilities in **vgrindefs** are of two types: Boolean capabilities which indicate that the language has some particular feature and string capabilities which give a regular expression or keyword list. Entries may continue onto multiple lines by giving a \ as the last character of a line. Lines starting with # are comments.

Capabilities

The following table names and describes each capability.

Name	Type	Description
ab	str	Regular expression for the start of an alternate form comment
ae	str	Regular expression for the end of an alternate form comment
bb	str	Regular expression for the start of a block
be	str	Regular expression for the end of a lexical block
cb	str	Regular expression for the start of a comment
ce	str	Regular expression for the end of a comment
id	str	String giving characters other than letters and digits that may legally occur in identifiers (default '_')
kw	str	A list of keywords separated by spaces
lb	str	Regular expression for the start of a character constant
le	str	Regular expression for the end of a character constant
oc	bool	Present means upper and lower case are equivalent
pb	str	Regular expression for start of a procedure
pl	bool	Procedure definitions are constrained to the lexical level matched by the 'px' capability
px	str	A match for this regular expression indicates that procedure definitions may occur at the next lexical level. Useful for lisp-like languages in which procedure definitions occur as subexpressions of defuns.
sb	str	Regular expression for the start of a string
se	str	Regular expression for the end of a string
tc	str	Use the named entry as a continuation of this one
tl	bool	Present means procedures are only defined at the top lexical level

Regular Expressions

vgrindefs uses regular expressions similar to those of **ex**(1) and **lex**(1). The characters '^', '$', ':', and '\' are reserved characters and must be 'quoted' with a preceding \ if they are to be included as normal characters. The metasymbols and their meanings are:

$	The end of a line
^	The beginning of a line
\d	A delimiter (space, tab, newline, start of line)
\a	Matches any string of symbols (like '.*' in lex)
\p	Matches any identifier. In a procedure definition (the 'pb' capability) the string that matches this symbol is used as the procedure name.
()	Grouping
|	Alternation
?	Last item is optional
\e	Preceding any string means that the string will not match an input string if the input string is preceded by an escape character (\). This is typically used for languages (like C) that can include the string delimiter in a string by escaping it.

Unlike other regular expressions in the system, these match words and not characters. Hence something like '(tramp|steamer)flies?' would match 'tramp', 'steamer', 'trampflies', or 'steamerflies'. Contrary to some forms of regular expressions, **vgrindef** alternation binds very tightly. Grouping parentheses are likely to be necessary in expressions involving alternation.

Keyword List

The keyword list is just a list of keywords in the language separated by spaces. If the 'oc' boolean is specified, indicating that upper and lower case are equivalent, then all the keywords should be specified in lower case.

EXAMPLE

The following entry, which describes the C language, is typical of a language entry.

```
C|c|the C programming language:\
        :pb=^\d?*?\d?\p\d??):bb={:be=}:cb=/*:ce=*/:sb=":se=\e":\
        :lb=':le=\e':tl:\
        :kw=asm auto break case char continue default do double else enum\
        extern float for fortran goto if int long register return short\
        sizeof static struct switch typedef union unsigned while #define\
        #else #endif #if #ifdef #ifndef #include #undef # define else endif\
        if ifdef ifndef include undef:
```

Note that the first field is just the language name (and any variants of it). Thus the C language could be specified to **vgrind**(1) as 'c' or 'C'.

FILES

/usr/lib/vgrindefs file containing terminal descriptions

SEE ALSO

troff(1), **vgrind**(1)

NAME

ypaliases – NIS aliases for sendmail

SYNOPSIS

/etc/ypaliases

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

Create the Network Information Service (NIS) aliases map with this text file. The **/etc/ypaliases** file has the same format as the **/etc/aliases** file described in **aliases**(5).

The text file for the NIS aliases map is stored in the **/etc/aliases** file on the NIS master of an NIS domain. Other systems in a domain (besides the NIS master) can also have a local **/etc/aliases** file. The local file is accessed first by programs such as **sendmail**(8), and if it contains a line beginning with the character '+', the NIS map will be accessed.

The local **/etc/aliases** file can specify resources that are not available on a network-wide basis. This implies that the NIS master cannot use the local **/etc/aliases** file to specify aliases that are to be known only to the local system. Sun386i systems allow the **/etc/aliases** file on the NIS master to be used locally, creating the NIS aliases map with the **/etc/ypaliases** text file.

FILES

/etc/aliases
/etc/ypaliases

SEE ALSO

uucp(1C), **dbm**(3X), **aliases**(5), **newaliases**(8), **sendmail**(8)

System and Network Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypfiles – NIS database and directory structure

DESCRIPTION

The Network Information Service (NIS) uses a distributed, replicated database of **dbm** files contained in the **/var/yp** directory hierarchy on each NIS server. A **dbm** database consists of two files, created by calls to the **ndbm**(3) library package. One has the filename extension **.pag** and the other has the filename extension **.dir**. For instance, the database named **hosts.byname**, is implemented by the pair of files **hosts.byname.pag** and **hosts.byname.dir**.

A **dbm** database served by the NIS service is called an NIS *map*. An NIS *domain* is a subdirectory of **/var/yp** containing a set of NIS maps. Any number of NIS domains can exist. Each may contain any number of maps.

No maps are required by the NIS lookup service itself, although they may be required for the normal operation of other parts of the system. There is no list of maps which the NIS service serves — if the map exists in a given domain, and a client asks about it, the NIS service will serve it. For a map to be accessible consistently, it must exist on all NIS servers that serve the domain. To provide data consistency between the replicated maps, an entry to run **ypxfr** periodically should be made in the super-user's **crontab** file on each server. More information on this topic is in **ypxfr**(8).

The NIS maps should contain two distinguished key-value pairs. The first is the key **YP_LAST_MODIFIED**, having as a value a ten-character ASCII order number. The order number should be the system time in seconds when the map was built. The second key is **YP_MASTER_NAME**, with the name of the NIS master server as a value. **makedbm**(8) generates both key-value pairs automatically. A map that does not contain both key-value pairs can be served by the NIS service, but the **ypserv** process will not be able to return values for "Get order number" or "Get master name" requests. See **ypserv**(8). In addition, values of these two keys are used by **ypxfr** when it transfers a map from a master NIS server to a slave. If **ypxfr** cannot figure out where to get the map, or if it is unable to determine whether the local copy is more recent than the copy at the master, you must set extra command line switches when you run it.

The NIS maps must be generated and modified only at the master server. They are copied to the slaves using **ypxfr**(8) to avoid potential byte-ordering problems among the NIS servers running on machines with different architectures, and to minimize the amount of disk space required for the dbm files. The NIS database can be initially set up for both masters and slaves by using **ypinit**(8).

After the server databases are set up, it is probable that the contents of some maps will change. In general, some ASCII source version of the database exists on the master, and it is changed with a standard text editor. The update is incorporated into the NIS map and is propagated from the master to the slaves by running **/var/yp/Makefile**. All Sun-supplied maps have entries in **/var/yp/Makefile**; if you add an NIS map, edit this file to support the new map. The makefile uses **makedbm**(8) to generate the NIS map on the master, and **yppush**(8) to propagate the changed map to the slaves. **yppush** is a client of the map **ypservers**, which lists all the NIS servers. For more information on this topic, see **yppush**(8).

FILES

/var/yp
/var/yp/Makefile

SEE ALSO

dbm(3X), **makedbm**(8), **rpcinfo**(8C), **ypinit**(8), **ypmake**(8), **yppoll**(8), **yppush**(8), **ypserv**(8), **ypxfr**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypgroup – NIS group file

SYNOPSIS

/etc/ypgroup

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

Create the Network Information Service (NIS) group map with this text file. This file has the same format as the **/etc/group** file described in **group**(5).

The text file for the NIS group map is stored in the **/etc/group** file on the NIS master of an NIS domain. Other systems in a domain (besides the NIS master) can also have a local **/etc/group** file. The local file is accessed first by programs such as **groups**(1), and if it contains a line beginning with the character '+', the NIS map will be accessed. The local **/etc/group** file can specify groups that are not available on a network-wide basis.

This implies that the NIS master cannot use the local **/etc/group** file to specify groups that are to be known only to the local system. Sun386i systems allow the **/etc/group** file on the NIS master to be used locally, creating the NIS group map from the **/etc/ypgroup** text file.

FILES

/etc/group
/etc/ypgroup

SEE ALSO

passwd(1), **su**(1V), **getgroups**(2V), **crypt**(3), **initgroups**(3), **group**(5), **group.adjunct**(5), **passwd**(5), **grpck**(8V)

System and Network Administration,
Sun386i SNAP Administration,
Sun386i Advanced Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

yppasswd – NIS password file

SYNOPSIS

/etc/yppasswd

DESCRIPTION

Create the Network Information Service (NIS) password map with this text file. The format for **/etc/yppasswd** is the same as for the **/etc/passwd** file described in **passwd**(5).

The text file for the NIS password map is stored in the **/etc/passwd** file on the NIS master of an NIS domain. Other systems in a domain can also have a local **/etc/passwd** file. The local file is accessed first by programs such as **passwd**(1), and if it contains a line beginning with the character '+', the NIS map will be accessed.

The local **/etc/passwd** file can specify users that are not available on a network-wide basis. This implies that the NIS master cannot use the local **/etc/passwd** file to specify users that are to be known only to the local system. Sun386i systems allow the **/etc/passwd** file on the NIS master to be used locally, creating the NIS password map from the **/etc/yppasswd** text file.

FILES

/etc/passwd
/etc/yppasswd

SEE ALSO

login(1), **mail**(1), **passwd**(1), **crypt**(3), **getpwent**(3V), **group**(5), **passwd**(5), **passwd.adjunct**(5), **adduser**(8), **sendmail**(8), **vipw**(8)

System and Network Administration,
Sun386i SNAP Administration,
Sun386i Advanced Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypprintcap – NIS printer capability database

SYNOPSIS

/etc/ypprintcap

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

Create the Network Information Service (NIS) printcap map with this text file to centralize and simplify printer administration. The **/etc/ypprintcap** file has the same format as the **/etc/printcap** file described in **printcap**(5).

The text file for the NIS printcap map is stored in the **/etc/printcap** file on the NIS master of an NIS domain. Other systems in a domain (besides the NIS master) can also have a local **/etc/printcap** file. The local file is accessed first by programs such as **lpr**(1), and if it contains a line beginning with the character '+', the NIS map will be accessed.

The local **/etc/printcap** file can specify printers that are not available on a network-wide basis. This implies that the NIS master cannot use the local **/etc/printcap** file to specify printers that are to be known only to the local system. Sun386i systems allow the **/etc/printcap** file on the NIS master to be used locally, using the **/etc/ypprintcap** file to create the NIS printcap map.

FILES

/etc/printcap
/etc/ypprintcap

SEE ALSO

lpq(1), **lpr**(1), **lprm**(1), **snap**(1), **stty**(1V), **plot**(3X), **ttcompat**(4M), **printcap**(5), **termcap**(5), **lpc**(8), **lpd**(8), **pac**(8)

System and Network Administration,
Sun386i SNAP Administration,
Sun386i Advanced Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

intro – introduction to games and demos

DESCRIPTION

This section describes available games and demos.

LIST OF GAMES AND DEMOS

Name	Appears on Page	Description
adventure	**adventure**(6)	an exploration game
arithmetic	**arithmetic**(6)	provide drill in number facts
backgammon	**backgammon**(6)	the game of backgammon
banner	**banner**(6)	print large banner on printer
battlestar	**battlestar**(6)	a tropical adventure game
bcd	**bcd**(6)	convert to antique media
bdemos	**bdemos**(6)	demonstrate Sun Monochrome Bitmap Display
bdraw	**draw**(6)	interactive graphics drawing
bj	**bj**(6)	the game of black jack
boggle	**boggle**(6)	play the game of boggle
boggletool	**boggletool**(6)	play a game of boggle
bouncedemo	**graphics_demos**(6)	graphics demonstration programs
brotcube	**brotcube**(6)	rotate a simple cube
bsuncube	**bsuncube**(6)	view 3-D Sun logo
buttontest	**buttontest**(6)	demonstration and testing program for SunButtons
canfield	**canfield**(6)	Canfield solitaire card game
canfieldtool	**canfield**(6)	Canfield solitaire card game
canvas_demo	**sunview_demos**(6)	Window-System demonstration programs
cdplayer	**cdplayer**(6)	CD-ROM audio demo program
cdraw	**draw**(6)	interactive graphics drawing
cfscores	**canfield**(6)	Canfield solitaire card game
chess	**chess**(6)	the game of chess
chesstool	**chesstool**(6)	window-based front-end to chess program
ching	**ching**(6)	the book of changes and other cookies
colordemos	**colordemos**(6)	demonstrate Sun Color Graphics Display
craps	**craps**(6)	the game of craps
cribbage	**cribbage**(6)	the card game cribbage
cursor_demo	**sunview_demos**(6)	Window-System demonstration programs
dialtest	**dialtest**(6)	demonstration and testing program for SunDials
draw	**draw**(6)	interactive graphics drawing
factor	**factor**(6)	factor a number, generate large primes
fish	**fish**(6)	play "Go Fish"
flight	**gp_demos**(6)	demonstration programs for the Graphics Processor
fortune	**fortune**(6)	print a random, hopefully interesting, adage
framedemo	**graphics_demos**(6)	graphics demonstration programs
gaintool	**gaintool**(6)	audio control panel
gammontool	**gammontool**(6)	play a game of backgammon
gp_demos	**gp_demos**(6)	demonstration programs for the Graphics Processor
graphics_demos	**graphics_demos**(6)	graphics demonstration programs
hack	**hack**(6)	replacement for rogue
hangman	**hangman**(6)	computer version of the game hangman
hunt	**hunt**(6)	a multiplayer multiterminal game
jumpdemo	**graphics_demos**(6)	graphics demonstration programs
life	**life**(6)	John Conway's game of life
mille	**mille**(6)	play Mille Bornes
monop	**monop**(6)	Monopoly game

moo	**moo**(6)	guessing game
number	**number**(6)	convert Arabic numerals to English
play	**play**(6)	play audio files
ppt	**bcd**(6)	convert to antique media
primes	**factor**(6)	factor a number, generate large primes
primes	**primes**(6)	print all primes larger than some given number
quiz	**quiz**(6)	test your knowledge
rain	**rain**(6)	animated raindrops display
random	**random**(6)	select lines randomly from a file
raw2audio	**raw2audio**(6)	convert raw audio data to audio file format
record	**record**(6)	record an audio file
robots	**robots**(6)	fight off villainous robots
rotcvph	**rotcvph**(6)	rotate convex polyhedron
rotobj	**gp_demos**(6)	demonstration programs for the Graphics Processor
snake	**snake**(6)	display chase game
snscore	**snake**(6)	display chase game
soundtool	**soundtool**(6)	audio play/record tool
spheresdemo	**graphics_demos**(6)	graphics demonstration programs
suncoredemos	**suncoredemos**(6)	demonstrate SunCore Graphics Package
sunview_demos	**sunview_demos**(6)	Window-System demonstration programs
trek	**trek**(6)	trekkie game
vwcvph	**vwcvph**(6)	view convex polyhedron
worm	**worm**(6)	play the growing worm game
worms	**worms**(6)	animate worms on a display terminal
wump	**wump**(6)	the game of hunt the wumpus

NAME

adventure – an exploration game

SYNOPSIS

/usr/games/adventure

DESCRIPTION

The object of the game is to locate and explore Colossal Cave, find the treasures hidden there, and bring them back to the building with you. The program is self-describing to a point, but part of the game is to discover its rules.

To terminate a game, type **quit**; to save a game for later resumption, type **suspend**.

BUGS

Saving a game creates a large executable file instead of just the information needed to resume the game.

NAME

arithmetic – provide drill in number facts

SYNOPSIS

/usr/games/arithmetic [+−x/] [*range*]

DESCRIPTION

arithmetic types out simple arithmetic problems, and waits for an answer to be typed in. If the answer is correct, it types back "Right!", and a new problem. If the answer is wrong, it replies "What?", and waits for another answer. Every twenty problems, it publishes statistics on correctness and the time required to answer.

To quit the program, type an interrupt (such as CTRL–C).

The first optional argument determines the kind of problem to be generated; '+', '−', 'x', '/' respectively cause addition, subtraction, multiplication, and division problems to be generated. One or more characters can be given; if more than one is given, the different types of problems will be mixed in random order; default is +−.

range is a decimal number; all addends, subtrahends, differences, multiplicands, divisors, and quotients will be less than or equal to the value of *range*. Default *range* is 10.

At the start, all numbers less than or equal to *range* are equally likely to appear. If the respondent makes a mistake, the numbers in the problem which was missed become more likely to reappear.

As a matter of educational philosophy, the program will not give correct answers, since the learner should, in principle, be able to calculate them. Thus the program is intended to provide drill for someone just past the first learning stage, not to teach number facts *de novo*. For almost all users, the relevant statistic should be time per problem, not percent correct.

NAME

backgammon – the game of backgammon

SYNOPSIS

backgammon [–] [**n r w b pr pw pb t***term* **s***filename*]

DESCRIPTION

backgammon lets you play backgammon against the computer or against a 'friend'. All commands only are one letter, so you don't need to type a carriage return, except at the end of a move. **backgammon** is mostly self documenting, so that a **q ?** (question mark) will usually get some help. If you answer **y** when **backgammon** asks if you want the rules, you will get text explaining the rules of the game, some hints on strategy, instruction on how to use **backgammon,** and a tutorial consisting of a practice game against the computer. A description of how to use **backgammon** can be obtained by answering **y** when it asks if you want instructions. The possible arguments for **backgammon** (most are unnecessary but some are very convenient) consist of:

n	don't ask for rules or instructions
r	player is red (implies n)
w	player is white (implies n)
b	two players, red and white (implies n)
pr	print the board before red's turn
pw	print the board before white's turn
pb	print the board before both player's turn
t*term*	terminal is type *term*, uses */etc/termcap*, otherwise uses the TERM environment variable.
s*filename*	recover previously saved game from *filename*. This can also be done by executing the saved file, that is, typing its name in as a command.

Arguments may be optionally preceded by a – sign. Several arguments may be concatenated together, but not after **s** or **t** arguments, since they can be followed by an arbitrary string. Any unrecognized arguments are ignored. An argument of a lone – gets a description of possible arguments.

If **term** has capabilities for direct cursor movement. **backgammon** 'fixes' the board after each move, so the board does not need to be reprinted, unless the screen suffers some horrendous malady. Also, any 'p' option will be ignored.

QUICK REFERENCE

When **backgammon** prompts by typing only your color, type a space or carriage return to roll, or

d	to double
p	to print the board
q	to quit
s	to save the game for later

When **backgammon** prompts with 'Move:', type

p	to print the board
q	to quit
s	to save the game

or a *move*, which is a sequence of

s-f	move from **s** to **f**
s/r	move one man on **s** the roll **r** separated by commas or spaces and ending with a newline. Available abbreviations are

s-f1-f2 means **s-f1,f1-f2**

s/r1r2 means **s/r1,s/r2**

Use **b** for bar and **h** for home, or **0** or **25** as appropriate.

FILES

/usr/games/teachgammon	rules and tutorial
/etc/termcap	terminal capabilities

BUGS

backgammon's strategy needs much work.

NAME

banner – print large banner on printer

SYNOPSIS

/usr/games/banner [**–w***n*] message ...

DESCRIPTION

banner prints a large, high quality banner on the standard output. If the message is omitted, it prompts for and reads one line of its standard input. If **–w** is given, the output is reduced from a width of 132 to *n*, suitable for a narrow terminal. If *n* is omitted, it defaults to 80.

The output should be printed on a hard-copy device, up to 132 columns wide, with no breaks between the pages. The volume is enough that you want a printer or a fast hardcopy terminal, but if you are patient, a decwriter or other 300 baud terminal will do.

BUGS

Several ASCII characters are not defined, notably '<', '>', '[', ']', '\', '^', '_', '{', '}', '|', and '~'. Also, the characters '"', ''', and '&' are funny looking (but in a useful way.)

The **–w** option is implemented by skipping some rows and columns. The smaller it gets, the grainier the output. Sometimes it runs letters together.

NAME

battlestar – a tropical adventure game

SYNOPSIS

battlestar [**–r**]

DESCRIPTION

battlestar is an adventure game in the classic style. However, it is slightly less of a puzzle and more a game of exploration. There are a few magical words in the game, but on the whole, simple English should suffice to make one's desires understandable to the parser.

OPTIONS

–r Recover a saved game.

THE SETTING

In the days before the darkness came, when battlestars ruled the heavens...

Three He made and gave them to His daughters,
Beautiful nymphs, the goddesses of the waters.
One to bring good luck and simple feats of wonder,
Two to wash the lands and churn the waves asunder,
Three to rule the world and purge the skies with thunder.

In those times great wizards were known and their powers were beyond belief. They could take any object from thin air, and, uttering the word '**su**', could disappear.

In those times men were known for their lust of gold and desire to wear fine weapons. Swords and coats of mail were fashioned that could withstand a laser blast.

But when the darkness fell, the rightful reigns were toppled. Swords and helms and heads of state went rolling across the grass. The entire fleet of battlestars was reduced to a single ship.

USAGE

Sample Commands

take	---	take an object
drop	---	drop an object
wear	---	wear an object you are holding
draw	---	carry an object you are wearing
puton	---	take an object and wear it
take off	---	draw an object and drop it

throw <object> <direction>
! <shell esc>

Implied Objects

```
>-: take watermelon
watermelon:
Taken.
>-: eat
watermelon:
Eaten.
>-: take knife and sword and apple, drop all
knife:
Taken.
broadsword:
Taken.
apple:
Taken.
knife:
Dropped.
```

```
broadsword:
Dropped.
apple:
Dropped.
>-: get
knife:
Taken.
```

Notice that the "shadow" of the next word stays around if you want to take advantage of it. That is, saying '**take knife**' and then '**drop**' will drop the knife you just took.

Score and Inven

The two commands **score** and **inven** will print out your current status in the game.

Saving a Game

The command **save** will save your game in a file called **Bstar**. You can recover a saved game by using the **–r** option when you start up the game.

Directions

The compass directions N, S, E, and W can be used if you have a compass. If you do not have a compass, you will have to say **R**, **L**, **A**, or **B**, which stand for Right, Left, Ahead, and Back. Directions printed in room descriptions are always printed in R, L, A, & B relative directions.

BUGS

Countless.

NAME

bcd, ppt – convert to antique media

SYNOPSIS

/usr/games/bcd *text*

/usr/games/ppt

DESCRIPTION

bcd converts the literal *text* into a form familiar to old-timers.

ppt converts the standard input into yet another form.

SEE ALSO

dd(1)

NAME

bdemos – demonstrate Sun Monochrome Bitmap Display

SYNOPSIS

/usr/demo/bballs
/usr/demo/bbounce
/usr/demo/bdemos
/usr/demo/bjump
/usr/demo/bphoto *file*
/usr/demo/brotcube

DESCRIPTION

Bdemos is a collection of simple demonstration programs for the Sun Monochrome Bitmap Display. Each program is briefly described below. Unless otherwise noted, each program should be terminated by typing the appropriate key (usually DELETE or ^C) to generate an interrupt signal.

bballs colliding balls demo

bbounce bouncing square demo

bdemos a collection of demos

This program has a menu for selection of several different demos. After typing a key to select a particular demo, the user may type ^C to get back the menu. Type 'q' to quit.

bjump simulated jump to hyperspace

bphoto *file* dither monochrome image *file* to bitmap display

Image files suitable for display by this program are in */usr/demo/bwpix*.

brotcube black and white spinning cube

FILES

/usr/demo/bwpix

SEE ALSO

bsuncube(6), **draw**(6)

NAME

bj – the game of black jack

SYNOPSIS

/usr/games/bj

DESCRIPTION

bj is a serious attempt at simulating the dealer in the game of black jack (or twenty-one) as might be found in Reno. The following rules apply:

The bet is $2 every hand.

A player "natural" (black jack) pays $3. A dealer natural loses $2. Both dealer and player naturals is a "push" (no money exchange).

If the dealer has an ace up, the player is allowed to make an "insurance" bet against the chance of a dealer natural. If this bet is not taken, play resumes as normal. If the bet is taken, it is a side bet where the player wins $2 if the dealer has a natural and loses $1 if the dealer does not.

If the player is dealt two cards of the same value, he is allowed to "double". He is allowed to play two hands, each with one of these cards. (The bet is doubled also; $2 on each hand.)

If a dealt hand has a total of ten or eleven, the player may "double down". He may double the bet ($2 to $4) and receive exactly one more card on that hand.

Under normal play, the player may "hit" (draw a card) as long as his total is not over twenty-one. If the player "busts" (goes over twenty-one), the dealer wins the bet.

When the player "stands" (decides not to hit), the dealer hits until he attains a total of seventeen or more. If the dealer busts, the player wins the bet.

If both player and dealer stand, the one with the largest total wins. A tie is a push.

The machine deals and keeps score. The following questions will be asked at appropriate times. Each question is answered by **y** followed by a new-line for "yes", or just new-line for "no".

? (this means, "do you want a hit?")
Insurance?
Double down?

Every time the deck is shuffled, the dealer so states and the "action" (total bet) and "standing" (total won or lost) is printed. To exit, hit the interrupt key (CTRL–C) and the action and standing will be printed.

NAME

boggle – play the game of boggle

SYNOPSIS

/usr/games/boggle [+] [++]

AVAILABILITY

This game is available with the *Games* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

This program is intended for people wishing to sharpen their skills at Boggle (TM Parker Bros.). If you invoke the program with 4 arguments of 4 letters each, (*e.g.* **"boggle appl epie moth erhd"**) the program forms the obvious Boggle grid and lists all the words from **/usr/dict/words** found therein. If you invoke the program without arguments, it will generate a board for you, let you enter words for 3 minutes, and then tell you how well you did relative to **/usr/dict/words**.

The object of Boggle is to find, within 3 minutes, as many words as possible in a 4 by 4 grid of letters. Words may be formed from any sequence of 3 or more adjacent letters in the grid. The letters may join horizontally, vertically, or diagonally. However, no position in the grid may be used more than once within any one word. In competitive play amongst humans, each player is given credit for those of his words which no other player has found.

In interactive play, enter your words separated by spaces, tabs, or newlines. A bell will ring when there is 2:00, 1:00, 0:10, 0:02, 0:01, and 0:00 time left. You may complete any word started before the expiration of time. You can surrender before time is up by hitting 'break'. While entering words, your erase character is only effective within the current word and your line kill character is ignored.

Advanced players may wish to invoke the program with 1 or 2 +'s as the first argument. The first + removes the restriction that positions can only be used once in each word. The second + causes a position to be considered adjacent to itself as well as its (up to) 8 neighbors.

NAME

boggletool – play a game of boggle

SYNOPSIS

/usr/games/boggletool [*number*] [+[+]] [16-character *string*]

AVAILABILITY

This game is available with the *Games* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

boggletool allows you to play the game of Boggle (TM Parker Bros.) against the computer. The *number* argument specifies the time limit in minutes (the default is 3 minutes). If a 16 character long string is placed on the command line, it is interpreted as a Boggle board: the first four letters form the top row, the next four letters the second row, etc. If no letters are specified, a board is randomly rolled by the computer from a set of Boggle cubes. The +[+] argument is explained below under **Advanced Play .**

PLAYING THE GAME

Rules of the Game

The object of Boggle is to find as many words as possible in a 4 by 4 grid of letters within a certain time limit. Words may be formed from any sequence of 3 or more adjacent letters in the grid. The letters may join horizontally, vertically, or diagonally. Normally, no letter in the grid may be used more than once in a word (see **Advanced Play** for exceptions).

Playing the Game

When invoked, boggletool displays a grid of letters and an hourglass. To enter words, simply type in lower case letters to spell the word you want. Use any whitespace (SPACE, TAB, or NEWLINE) to finish a word. To correct mistakes you make, use BACKSPACE or DEL to delete the last character, or use CTRL-U to delete an entire word. **boggletool** verifies that words you enter are both in the grid and are valid English words. If you type in a character which would form a word which is not in the grid, the display will flash and the character you typed will not be echoed. When you type any whitespace to end the current word, **boggletool** will verify that the word is three or more letters long and that it appears in the dictionary. If the word you typed is illegal for either reason, the display will flash and you will have to either erase the word or change it. If you try to enter a valid word which you have already entered, the display will flash and the previous occurrence of the word will be highlighted. Again, you will have to erase the word before continuing. As you enter words, the "sand" in the hourglass will fall. At the end of the time limit, the display will flash and you will no longer be allowed to enter words. After a moment, the computer will display two lists of words: the words you found, and other words which also appear in the grid. To play another game, just type any capital letter (or use the pop-up menu).

Using the Menu

The pop-up menu is invoked by pressing the RIGHT mouse button. There are four items in it, and they work as follows.

Restart Game
: Create a new boggletool a new board, reset the timer, and allow you to start from scratch.

Restart Timer
: Allows you to cheat by reseting the hourglass timer to zero.

Give Up
: End the game and print the results immediately.

Quit Allows you to quit running the boggletool program. A prompt appears asking you to confirm the quit; when it does, click the LEFT mouse button to quit or the RIGHT mouse button to abort the quit.

Advanced Play

There are two options for advanced players. If a single + appears on the command line, letters in the grid may be reused. If two +'s are on the command line, letters may also be considered adjacent to themselves as well as to their neighbors. Although it is far easier to find words with these two options, there are also many more possible words in the grid and it is therefore difficult to find them all.

FILES

/usr/games/boggledict dictionary file for computer's words

NAME

brotcube – rotate a simple cube

SYNOPSIS

/usr/demo/brotcube

DESCRIPTION

brotcube rotates a skeletal outline of a cube consisting of 14 vectors. Using the SunCore Graphics Package, a 3-D projection is drawn on the Sun Monochrome Bitmap Display. Each rotation consists of 100 views.

This program gives an indication of the performance of the SunCore Graphics Package.

Type **q** to exit the program.

NAME

bsuncube – view 3-D Sun logo

SYNOPSIS

/usr/demo/bsuncube

DESCRIPTION

bsuncube allows the user to view a cube from various positions with hidden faces removed. The faces of the cube consist of the Sun logo. The viewing position is selected using the mouse. Using the SunCore Graphics Package, a 3-D projection is drawn on the Sun Monochrome Bitmap Display.

The program operates in two modes: **DisplayObject** mode and **SelectView** mode. The program starts in **DisplayObject** mode:

DisplayObject: The cube is displayed in 3-D perspective with hidden faces removed. Type **q** while in this mode to exit the program. Press RIGHT mouse button to switch to **SelectView** mode.

SelectView: Schematic projections of the outline of the cube are shown and the mouse is used to select a viewing position. Use LEFT mouse button to set *x* and MIDDLE mouse button to set *y* in the *Front View*. Use MIDDLE mouse button to set *z* in the *Top View*. Press RIGHT mouse button to switch to **DisplayObject** mode.

The view shown in **DisplayObject** mode is drawn using the conventions that the viewer is always looking from the viewing position toward the center of the cube and that the positive *y* axis on the screen is the projection of the positive *y* axis in 3-D cube coordinates.

NAME

buttontest – demonstration and testing program for SunButtons

SYNOPSIS

/usr/demo/BUTTONBOX/buttontest

DESCRIPTION

buttontest displays a window with thirty two buttons, corresponding to those on SunButtons. To determine if the button box has been set up correctly, select the **Diagnostic** button on the panel. If the button box is correctly interfaced, **buttonbox OK** is displayed, and pressing a button on the box highlights a button on the screen. If **No Response from Buttonbox** is displayed, repeat the button box install procedure.

NAME

canfield, canfieldtool, cfscores – Canfield solitaire card game

SYNOPSIS

/usr/games/canfield [**–ac**]

/usr/games/canfieldtool [**–ac**]

/usr/games/cfscores [**–ac**] [*username*]

AVAILABILITY

These games are available with the *Games* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

canfield can be played on any terminal. **canfieldtool** is the SunView version with attractive graphics.

If you have never played solitaire before, it is recommended that you consult a solitaire instruction book. In **canfield**, tableau cards may be built on each other downward in alternate colors. An entire pile must be moved as a unit in building. Top cards of the piles are available to be able to be played on foundations, but never into empty spaces.

Spaces must be filled from the stock. The top card of the stock also is available to be played on foundations or built on tableau piles. After the stock is exhausted, tableau spaces may be filled from the talon and the player may keep them open until he wishes to use them.

Cards are dealt from the hand to the talon by threes and this repeats until there are no more cards in the hand or the player quits. To have cards dealt onto the talon the player types **ht** for his move. Foundation base cards are also automatically moved to the foundation when they become available.

Canfieldtool

Once you understand the rules, **canfieldtool** is self-explanatory.

Canfield

The rules for betting are somewhat less strict than those used in the official version of the game. The initial deal costs $13. You may quit at this point or inspect the game. Inspection costs $13 and allows you to make as many moves as is possible without moving any cards from your hand to the talon. (The initial deal places three cards on the talon; if all these cards are used, three more are made available.) Finally, if the game seems interesting, you must pay the final installment of $26. At this point you are credited at the rate of $5 for each card on the foundation; as the game progresses you are credited with $5 for each card that is moved to the foundation. Each run through the hand after the first costs $5. The card counting feature costs $1 for each unknown card that is identified. If the information is toggled on, you are only charged for cards that became visible since it was last turned on. Thus the maximum cost of information is $34. Playing time is charged at a rate of $1 per minute. If the **–a** flag is specified, it prints out the canfield accounts for all users that have played the game since the database was set up.

OPTIONS

a Print out **canfield** accounts for all users that have played the game since the database was set up.

c Maintain card counting statistics on the bottom of the screen. When properly used this can greatly increase the chances of winning.

With no arguments, **cfscores** prints out the current status of your canfield account. If *username* is specified, it prints out the status of their account.

FILES

/usr/games/canfield the game itself
/usr/games/lib/cfscores the database of scores

BUGS

It is impossible to cheat.

NAME

cdplayer – CD-ROM audio demo program

SYNOPSIS

cdplayer [-d *device*] [*sunview options*]

AVAILABILITY

This demo is available with the *Games* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

cdplayer demonstrate the CD quality audio capability of the CD-ROM drive. It is a SunView program and plays any Audio Compact Discs. There are four panels in the window. The top panel displays the all the available tracks on the CD. The user can select the any tracks by clicking it with the left mouse button. The second panel contains the play, pause, stop and eject button. The third panel display the CD music address and track number. The bottom panel contains the volume control slider and close button.

Refer to the CD-ROM hardware documentation for connecting the speakers or head-phones to the drive.

OPTIONS

–d *device* Use *device* as the CD-ROM device, rather than **/dev/rsr0** the default CD-ROM device.

FILES

/dev/rsr0 CD-ROM raw file

SEE ALSO

sr(4)

NAME

chess – the game of chess

SYNOPSIS

/usr/games/chess

AVAILABILITY

This game is available for Sun-3 and Sun-4 systems with the *Games* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

chess is a computer program that plays class D chess. Moves may be given either in standard (descriptive) notation or in algebraic notation. The symbol '+' is used to specify check; '**o-o**' and '**o-o-o**' specify castling. To play black, type '**first**'; to print the board, type an empty line.

Each move is echoed in the appropriate notation followed by the program's reply.

DIAGNOSTICS

The most cryptic diagnostic is '**eh?**' which means that the input was syntactically incorrect.

FILES

/usr/games/lib/chess.book
book of opening moves

BUGS

Pawns may be promoted only to queens.

NAME

chesstool – window-based front-end to chess program

SYNOPSIS

/usr/games/chesstool [*chess_program*]

AVAILABILITY

This game is available for Sun-3 and Sun-4 systems, with the *Games* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

chesstool is a window-based front-end to the **chess**(6) program. Used without options, **chesstool** uses **/usr/games/chess**; you can designate any alternate program which uses the same command syntax as **chess**(6) with the *chess_program* argument.

When **chesstool** starts up, it displays a large window with three subwindows. The first subwindow displays messages '**Illegal move**', for example. The second subwindow is an options subwindow; options are described below. The final subwindow is a chessboard display with white and black pieces and two (advisory only) timekeeping clocks.

Make your moves with the mouse: select a piece by positioning the arrow cursor over the piece and pressing the left mouse button down, then drag the piece to the destination square, and release the button. The cursor will then turn to an hourglass icon while the system plays.

Items in the subwindow may be selected with either the left or middle mouse buttons. These options are:

Last Play Show the last play made.

Undo Undo your last move and the machine's response.

Once the game is over, it is not possible to restart it, so undo will update the board, but the game cannot be continued from that position.

Flash Flash when the machine has completed its move.

When this command is selected, a check mark will appear next to the word **Flash**. In flash mode, if **chesstool** is open, the piece moved by the system on its play will flash until you make your move. If **chesstool** is iconic, the entire icon will flash when the machine has made its move. Thus you can "Close" **chesstool** and be alerted when it's your turn to move. To turn flash mode off, select flash again.

Machine White Start a new game with the machine playing white.

Human White Start a new game with the machine playing black.

Quit Exit from **chesstool**.

There are two moves which are special: castling and capturing a pawn *en*passant. To castle, move the king only. The position of the rook will automatically be updated. Since the king moves two squares when castling, the move is unambiguous. To capture *en*passant, move the pawn to the square occupied by the opposing pawn which will be captured.

SEE ALSO

chess(6)

NAME

ching – the book of changes and other cookies

SYNOPSIS

/usr/games/ching [*hexagram*]

DESCRIPTION

The *I Ching* or *Book of Changes* is an ancient Chinese oracle that has been in use for centuries as a source of wisdom and advice.

The text of the *oracle* (as it is sometimes known) consists of sixty-four *hexagrams*, each symbolized by a particular arrangement of six straight (——) and broken (– –) lines. These lines have values ranging from six through nine, with the even values indicating the broken lines.

Each *hexagram* consists of two major sections. The **Judgement** relates specifically to the matter at hand (For instance, "It furthers one to have somewhere to go.") while the **Image** describes the general attributes of the *hexagram* and how they apply to one's own life ("Thus the superior man makes himself strong and untiring.").

When any of the lines has the value six or nine, it is a moving line; for any such line there is an appended judgement which becomes significant. Furthermore, the moving lines are inherently unstable and change into their opposites; a second *hexagram* (and thus an additional judgement) is formed.

Normally, one consults the oracle by fixing the desired question firmly in mind and then casting a set of changes (lines) using yarrow–stalks or tossed coins. The resulting *hexagram* will be the answer to the question.

Using an algorithm suggested by S. C. Johnson, this oracle simply reads a question from the standard input (up to an EOF) and hashes the individual characters in combination with the time of day, process ID and any other magic numbers which happen to be lying around the system. The resulting value is used as the seed of a random number generator which drives a simulated coin–toss divination. The answer is then piped through **nroff** for formatting and will appear on the standard output.

For those who wish to remain steadfast in the old traditions, the oracle will also accept the results of a personal divination using, for example, coins. To do this, cast the change and then type the resulting line values as an argument.

The impatient modern may prefer to settle for Chinese cookies; try **fortune**(6).

SEE ALSO

It furthers one to see the great man.

DIAGNOSTICS

The great prince issues commands,
Founds states, vests families with fiefs.
Inferior people should not be employed.

BUGS

Waiting in the mud
Brings about the arrival of the enemy.

If one is not extremely careful,
Somebody may come up from behind and strike him.
Misfortune.

NAME

colordemos – demonstrate Sun Color Graphics Display

SYNOPSIS

/usr/demo/cballs
/usr/demo/cdraw
/usr/demo/cphoto *file*
/usr/demo/cpipes
/usr/demo/cshowmap *file*
/usr/demo/csnow
/usr/demo/csuncube
/usr/demo/csunlogo
/usr/demo/cvlsi

DESCRIPTION

colordemos is a collection of simple demonstration programs for the Sun Color Graphics Display. Each program is briefly described below. To exit each program, send an interrupt signal by typing the appropriate key (usually CTRL-C).

cballs	Colliding balls on color display.
cdraw	Draw on the color display (see **draw**(6) for an explanation of how to use **cdraw**).
cphoto *file*	Display dithered color file on color display. Files suitable for display are in **/usr/demo/colorpix**.
cpipes	Colliding pipes on color display.
cshowmap *file*	Display maps. Files suitable for display are in **/usr/demo/segments**.
csnow	Color kaleidoscope.
csuncube	Multicolored Sun logo.
csunlogo	Shaded Sun logo.
cvlsi	Color VLSI layout demo.

FILES

/usr/demo/colorpix
/usr/demo/segments

NAME

craps – the game of craps

SYNOPSIS

/usr/games/craps

DESCRIPTION

craps is a form of the game of craps that is played in Las Vegas. The program simulates the *roller*, while the user (the *player*) places bets. The player may choose, at any time, to bet with the roller or with the *House*. A bet of a negative amount is taken as a bet with the House, any other bet is a bet with the roller.

The player starts off with a "bankroll" of $2,000.

The program prompts with:

bet?

The bet can be all or part of the player's bankroll. Any bet over the total bankroll is rejected and the program prompts with **bet?** until a proper bet is made.

Once the bet is accepted, the roller throws the dice. The following rules apply (the player wins or loses depending on whether the bet is placed with the roller or with the House; the odds are even). The *first* roll is the roll immediately following a bet:

1. On the first roll:

7 or 11	wins for the roller;
2, 3, or 12	wins for the House;
any other number	is the *point*, roll again (Rule 2 applies).

2. On subsequent rolls:

point	roller wins;
7	House wins;
any other number	roll again.

If a player loses the entire bankroll, the House will offer to lend the player an additional $2,000. The program will prompt:

marker?

A **yes** (or **y**) consummates the loan. Any other reply terminates the game.

If a player owes the House money, the House reminds the player, before a bet is placed, how many markers are outstanding.

If, at any time, the bankroll of a player who has outstanding markers exceeds $2,000, the House asks:

Repay marker?

A reply of **yes** (or **y**) indicates the player's willingness to repay the loan. If only 1 marker is outstanding, it is immediately repaid. However, if more than 1 marker are outstanding, the House asks:

How many?

markers the player would like to repay. If an invalid number is entered (or just a carriage return), an appropriate message is printed and the program will prompt with **How many?** until a valid number is entered.

If a player accumulates 10 markers (a total of $20,000 borrowed from the House), the program informs the player of the situation and exits.

Should the bankroll of a player who has outstanding markers exceed $50,000, the *total* amount of money borrowed will be *automatically* repaid to the House.

Any player who accumulates $100,000 or more breaks the bank. The program then prompts:

New game?

to give the House a chance to win back its money.

Any reply other than **yes** is considered to be a **no** (except in the case of **bet?** or **How many?**). To exit, send an interrupt (break), DELETE character or CTRL–D The program will indicate whether the player won, lost, or broke even.

MISCELLANEOUS

The random number generator for the die numbers uses the seconds from the time of day. Depending on system usage, these numbers, at times, may seem strange but occurrences of this type in a real dice situation are not uncommon.

NAME

cribbage – the card game cribbage

SYNOPSIS

/usr/games/cribbage [**–eqr**] *name* ...

DESCRIPTION

cribbage plays the card game cribbage, with **cribbage** playing one hand and the user the other. **cribbage** initially asks the user if the rules of the game are needed – if so, **cribbage** displays the appropriate section from *According to Hoyle* with **more**(1).

OPTIONS

–e Provide an explanation of the correct score when the player makes mistakes scoring his hand or crib. This is especially useful for beginning players.

–q Print a shorter form of all messages – this is only recommended for users who have played the game without specifying this option.

–r Instead of asking the player to cut the deck, **cribbage** will randomly cut the deck.

PLAYING CRIBBAGE

cribbage first asks the player whether he wishes to play a short game ("once around", to 61) or a long game ("twice around", to 121). A response of 's' results in a short game, any other response plays a long game.

At the start of the first game, **cribbage** asks the player to cut the deck to determine who gets the first crib. The user should respond with a number between 0 and 51, indicating how many cards down the deck is to be cut. The player who cuts the lower ranked card gets the first crib. If more than one game is played, the loser of the previous game gets the first crib in the current game.

For each hand, **cribbage** first prints the player's hand, whose crib it is, and then asks the player to discard two cards into the crib. The cards are prompted for one per line, and are typed as explained below.

After discarding, **cribbage** cuts the deck (if it is the player's crib) or asks the player to cut the deck (if it's its crib); in the latter case, the appropriate response is a number from 0 to 39 indicating how far down the remaining 40 cards are to be cut.

After cutting the deck, play starts with the non-dealer (the person who doesn't have the crib) leading the first card. Play continues, as per cribbage, until all cards are exhausted. **cribbage** keeps track of the scoring of all points and the total of the cards on the table.

After play, the hands are scored. **cribbage** requests the player to score his hand (and the crib, if it is his) by printing out the appropriate cards (and the cut card enclosed in brackets). Play continues until one player reaches the game limit (61 or 121).

A carriage return when a numeric input is expected is equivalent to typing the lowest legal value; when cutting the deck this is equivalent to choosing the top card.

SPECIFYING CARDS

Cards are specified as *rank* followed by *suit*. The *rank*s may be specified as one of **a**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **t**, **j**, **q**, and **k**, or alternatively, one of **ace**, **two**, **three**, **four**, **five**, **six**, **seven**, **eight**, **nine**, **ten**, **jack**, **queen**, and **king**. *Suit*s may be specified as **s**, **h**, **d**, and **c**, or alternatively as **spades**, **hearts**, **diamonds**, and **clubs**. A card may be specified as *rank suit*, or *rank* of *suit*. If the single letter *rank* and *suit* designations are used, the space separating the *suit* and *rank* may be left out. Also, if only one card of the desired *rank* is playable, typing the *rank* is sufficient. For example, if your hand was **2h**, **4d**, **5c**, **6h**, **jc**, **kd** and you wanted to discard the king of diamonds, you could type any of **k**, **king**, **kd**, **k d**, **k of d**, **king d**, **king of d**, **k diamonds**, **k of diamonds**, **king diamonds**, or **king of diamonds**,

FILES

/usr/games/cribbage

SEE ALSO

more(1)

NAME

dialtest – demonstration and testing program for SunDials

SYNOPSIS

/usr/demo/DIALBOX/dialtest

DESCRIPTION

dialtest displays a window with eight dials, corresponding to those on SunDials. To determine if the dialbox has been set up correctly, select the **Diagnostic** button on the panel. If the dialbox is correctly interfaced, **Dialbox OK** is displayed, and turning a dial on the box turn a dial on the screen. If **No Response from Dialbox** is displayed, repeat the dialbox install procedure.

NAME

draw, bdraw, cdraw – interactive graphics drawing

SYNOPSIS

/usr/demo/bdraw
/usr/demo/cdraw

DESCRIPTION

The *draw* programs are menu-driven programs which use the mouse, keyboard, bitmap display and optionally the color display to draw objects, drag them around, save them on disk, and so on. **bdraw** is the draw program for the black and white display and **cdraw** is the program for driving the color display.

The main menu items are selected by moving the mouse cursor and pressing the left mouse button. To redraw the display, point at the left edge of the main menu box and press the left button. The main menu items are:

New Seg xlate
Open a new translatable segment. A segment is a collection of attributes and primitives (lines, text, polygons, etc.). A translatable segment may subsequently be positioned.

New Seg xform
Open a new transformable segment. A transformable segment may subsequently be rotated, scaled, or positioned.

Delete Seg To delete a segment, point at any primitive in the segment and press the left button.

Lines To add line primitives to the currently open segment, position cursor, press the left button, ... press right button to quit.

Polygon To add a polygon primitive to the currently open segment, position the cursor, press the left button, ... press the right button to terminate the boundary definition. Polygons are filled with the current fill attribute.

Raster To add a raster primitive to the currently open segment, position the cursor, press the left button to reposition the box, adjust the box by moving the mouse, press the right button to create the raster primitive comprising the boxed bitmap. A 'rasterfile' is also created on disk for hardcopy purposes (see */usr/include/rasterfile.h*). This 'rasterfile' file may be spooled to a Versatec printer/plotter for hardcopy after exiting from the draw program. The command to do this is **lpr −v rasterfile**.

Text To add a text primitive to the currently open segment, position cursor, press left button, type the text string at the keyboard (back space works), hit return. Text is drawn with the current text attributes.

Marker To add marker primitives to the currently open segment, position cursor, press the left button to place marker, ... press the right button to quit.

Position To position a segment, point at any primitive in the segment, press left button, position the segment, press right button to quit.

Rotate To rotate a transformable segment, point at any primitive in the segment, press left button, move mouse to rotate, press right button to quit.

Scale To scale a transformable segment, point at any primitive in the segment, press the left button, move mouse to scale in x or y, press right button to quit.

Attributes This item brings up the attribute menu. To select an attribute such as text font, region fill texture (color), linestyle, or line width, point at the item and press the left button. Point at the left edge of the menu box to quit.

Save Seg To save a segment on a disk file, point at the segment, press the left button, type the disk file name, hit return.

Restore Seg
To restore a previously saved segment from disk, type file name, hit return.

Exit Exit the draw program.

BUGS

Rasters and raster text do not scale or rotate. If segments completely overlap, only the last one drawn may be picked by pointing with the mouse. This also applies to the menu segments! Therefore, don't cover them up with polygons. If aborted with your interrupt character, you must give the 'reset' command to turn keyboard echo back on and to reset -cbreak. Therefore, use the Exit item in the main menu to exit the program.

NAME

factor, primes – factor a number, generate large primes

SYNOPSIS

/usr/games/factor [*number*]

/usr/games/primes [*number*]

DESCRIPTION

factor reads lines from its standard input. If it reads a positive number, **factor** will factor the number and print its prime factors, printing each one the proper number of times. **factor** exits when it reads zero, a negative number, or something other than a number. If a *number* is given, **factor** will factor the number, print its prime factors, and exit.

primes reads a number from the standard input and prints all primes larger than the given number and smaller than 2^{32} (about 4.3×10^9). If a *number* is given, **primes** will use that number rather than reading one from the standard input.

DIAGNOSTICS

Ouch. Input out of range or for garbage input.

NAME

fish – play ''Go Fish''

SYNOPSIS

/usr/games/fish

DESCRIPTION

fish plays the game of "Go Fish", a children's card game. The object is to accumulate "books" of 4 cards with the same face value. The players alternate turns; each turn begins with one player selecting a card from his hand, and asking the other player for all cards of that face value. If the other player has one or more cards of that face value in his hand, he gives them to the first player, and the first player makes another request. Eventually, the first player asks for a card which is not in the second player's hand: he replies 'GO FISH!' The first player then draws a card from the "pool" of undealt cards. If this is the card he had last requested, he draws again. When a book is made, either through drawing or requesting, the cards are laid down and no further action takes place with that face value.

To play the computer, simply make guesses by typing **a**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**, **j**, **q**, or **k** when asked. Hitting a RETURN character gives you information about the size of my hand and the pool, and tells you about my books. Saying '**p**' as a first guess puts you into "pro" level; the default is pretty dumb.

NAME

fortune – print a random, hopefully interesting, adage

SYNOPSIS

/usr/games/fortune [–] [**–alsw**] [*filename*]

DESCRIPTION

fortune with no arguments prints out a random adage. The flags mean:

–a Choose from either list of adages.

–l Long messages only.

–s Short messages only.

–w Waits before termination for an amount of time calculated from the number of characters in the message. This is useful if it is executed as part of the logout procedure to guarantee that the message can be read before the screen is cleared.

FILES

/usr/games/lib/fortunes.dat

NAME

gaintool – audio control panel

SYNOPSIS

gaintool

AVAILABILITY

This command is only available with the *Demos* installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

gaintool is a SunView demonstration program that controls various characteristics of the SPARCstation 1 audio device, see **audio**(4S). Operations performed by **gaintool** affect all audio programs; for instance, adjusting the **Play Volume** instantly changes the output gain, regardless of which program is playing. **gaintool** also detects audio state changes made by other programs, and updates its display accordingly, keeping **gaintool** in sync with the current device configuration.

gaintool demonstrates an important principle involved in the integration of audio in the desktop environment: by enabling global control of important characteristics, it is not necessary for every application to provide an interface for these parameters. For instance, since audio output may be paused from the control panel, it is not strictly necessary that output applications display a **Pause** button of their own. However, such applications may detect that audio output has been paused, and take appropriate action.

Control Panel

Play Volume

This slider adjusts the output volume. Volume levels between 0 and 100 may be selected, where 0 represents infinite attenuation and 100 is maximum gain.

Record Volume

This slider adjusts the recording gain level in the range 0 to 100.

Monitor Volume

This slider adjusts the monitor gain level in the range 0 to 100. Monitor gain controls the amount of audio input signal that is fed through to the output port. For instance, if an audio source (such as a radio or CD-player) is connected directly to the input port, the input signal may be monitored through either the built-in speaker or the headphone jack.

Output This selector switches the audio output port between the built-in speaker and the external headphone jack.

Pause Play

This button may be used to suspend and resume audio output. If audio output is in progress when **Pause** is clicked, it is stopped immediately and subsequent output data remains queued. The button then switches to a **Resume** button that, when clicked, resumes audio output at the point that it was suspended.

If no process has the device open for output when **Pause** is clicked, **gaintool** holds the device open itself, thereby denying other processes output access. Audio programs that simply open and write to the audio device will typically be suspended when they attempt to open the device. Programs that asynchronously poll the device will discover that it is "busy" and may take appropriate action.

Audio Device Status Panel

Pressing the **PROPS** (L3) key brings up a status panel that shows the current state with the its display accordingly, audio applications. Selecting "Done" from the panel menu (or pressing the (L7) key) removes the panel.

Ordinarily, the device status is updated only when a **SIGPOLL** signal is delivered to **gaintool** (see **audio**(4S)). Because of this, the **Active** and **Samples** indicators are not necessarily kept up-to-date. However, when the mouse is positioned over the panel, status is continually updated.

SEE ALSO

audio(4S), **soundtool**(6)

BUGS

Record Volume should be controlled by a separate panel that also provides automatic gain level adjustment capabilities.

WARNINGS

This program is furnished on an *as is* basis as a demonstration of audio applications programming.

NAME

gammontool – play a game of backgammon

SYNOPSIS

/usr/games/gammontool [*path*]

AVAILABILITY

This game is available with the *Games* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

gammontool paints a backgammon board on the screen, and then lets you play against the computer. It must be run in SunWindows. The optional *path* argument specifies an alternate move-generating program, which must be specially designed to run with **gammontool**.

The game has three subwindows: an option window on top, a message window in the middle, and a large board on the bottom. The buttons in the option window are used to restart, double, etc. The message window has two lines: the first tells whose turn it is, and the second displays any errors that occur.

The Initial Roll

To start the game, roll the dice to determine who goes first. Move the mouse arrow onto the board and click the left button. One die appears on each side of the board: the die on the left is yours, and the die on the right is the computer's. If your roll is greater, then you move; if not, the computer makes a move.

Making Your Move

When it is your turn, '**Your**move' appears in the message window. Place the mouse over any piece of your color, and click the left button. While holding down the button, move the mouse to drag the piece; the piece follows the mouse until you release the button. The tool checks each move and does not allow illegal moves. When you have made as many moves as you can, the computer takes its turn; after it finishes, you may either roll again, or double.

Doubling

To double, click the *Double* button in the option window and wait for the computer's response. If the computer doubles you, a message is displayed and you must answer with the **Accept Double** or **Refuse Double** buttons. The **Forfeit** button can also be used to refuse a double. If the game is doubled, a doubling cube with the proper value is displayed on the bar strip. If the number is facing up, then you may double next. If the number is upside down, it is the computer's turn to double.

Other Buttons

If you want to change your move before you have finished it, use the **Redo Move** or **Redo Entire Move** buttons in the option window. **Redo Entire Move** replaces all of the pieces you have moved so that you can redo them all. **Redo Move** only replaces the last piece you moved, so it is useful when you roll doubles and want to redo only the last piece you moved. Note that once you have made all of the moves your roll permits, play passes immediately to the computer, so you cannot redo the very last move. The **Show Last Move** button allows you to see the last move again.

Leaving the Game

If you want to quit playing backgammon, use the **Quit** button. If you want to forfeit the game, use the **Forfeit** button. The computer penalizes you by taking a certain number of points, but the program does not terminate.

To play another game after winning, losing, or forfeiting, click the **New Game** button. To change the color of your pieces, click the mouse button while pointing at either the **White** or **Black** checkboxes. You may change colors at any time, even in the middle of a game. Changing colors in the middle of a game does not mean that you trade places with the computer; your pieces stay where they are, but they are repainted with the new color. Your pieces always move from the top right to the bottom right of the board, regardless of your color. As an additional cue as to your color, your dice are always displayed on the left half of the board.

Log File

If a there is a **gammonlog** file your home directory, **gammontool** keeps a log of the games played. Each move and double gets recorded, along with the winners and accumulated scores.

FILES

~/gammonlog	log of games played
/usr/games/lib/gammonscores	log of wins and losses

BUGS

The default strategy used by the computer is very poor.

If a single move uses more than one die (for instance if you roll 5, 6 and move 11 spaces without touching down in the middle) it is unpredictable where the program will make the piece touch down. This may be important if there is a blot on one of these middle points. The program will always make the move if possible, but if two midpoints would work and there is a blot on one of them, it is much better to explicitly hit the blot and then move the piece the rest of the way.

NAME

gp_demos, flight, rotobj – demonstration programs for the Graphics Processor

SYNOPSIS

/usr/demo/flight

/usr/demo/rotobj [*object*]

AVAILABILITY

These demos are available with the *Demos* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

These demos only run in windows running on a Graphics Processor surface.

Flight

flight is a mouse-driven flight simulator.

Interactive Commands

Middle-Button Restart the program.

Right-Button Increase speed.

Left-Button Decrease speed.

Move-Mouse-Forward
The airplane dives.

Move-Mouse-Backward
The airplane climbs.

Move-Mouse-Left/Right
The airplane banks.

Left/Right-With-Right-Button
The airplane rolls without banking.

Rotobj

rotobj rotates an *object*. Object files are located in **/usr/demo/DATA** and have the suffix **.vecs**.

FILES

/usr/demoDATA

SEE ALSO

graphics_demos(6)

NAME

graphics_demos, bouncedemo, framedemo, jumpdemo, spheresdemo, – graphics demonstration programs

SYNOPSIS

/usr/demo/bouncedemo [**–d** *dev*] [**–n***x*] [**–r**] [**–q**]

/usr/demo/framedemo [**–d** *dev*] [**–n***x*] [**–r**] [**–q**]

/usr/demo/jumpdemo [**–c**] [**–d** *dev*] [**–n***x*] [**–r**] [**–q**]

/usr/demo/spheresdemo [**–d** *dev*] [**–n***x*] [**–r**] [**–q**]

AVAILABILITY

These demos are available with the *Demos* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

bouncedemo

bouncedemo displays a bouncing square.

framedemo

framedemo

displays a series of frames, each of which contains a 256 by 256 image one-bit-deep pixels (that is, the image is a square monochrome bitmap, with 256 bits on a side). **framedemo** looks for the frames in the files **frame.1** through **frame.n** in the current working directory, and displays them in numerical order. A set of sample frames is available in the directory **/usr/demo/globeframes/***.

Interactive Commands

If you move the cursor onto the image surface, you can type certain commands to affect the rate at which the frames are displayed. The initial rate is one frame per second:

f Remove 1/20th of a second from the interval.

F Remove one second from the interval. **Ff** makes the interval as small as possible.

s Add 1/20th of a second.

S Add one second.

jumpdemo

jumpdemo simulates the famous **Star Wars** jump to light-speed-sequence using vector drawing. Colored stars are drawn on color surfaces.

spheresdemo

spheresdemo computes a random collection of shaded spheres. Colored spheres are drawn on color surfaces.

OPTIONS

–c Rotate the color map to produce a sparkling effect.

–d *surface*

Run the demo on a surface other than the window or system console, for instance:

bouncedemo –d /dev/cgone0

–n*x* Draw *x* items, or repeat a sequence *x* times.

–r Retain the window. This allows the image to reappear when uncovered instead of restarting the demo.

–q Quick exit. Useful for running several demos from within a shell script.

NAME

hack – replacement for rogue

SYNOPSIS

hack [**–d** *hackdir*] [**–s all** | *player* ...]

DESCRIPTION

hack is a display-oriented dungeons & dragons type game. Both display and command structure resemble *rogue*, although **hack** has twice as many monster types and requires three times as much memory.

Normally **hack** looks in **/usr/games/lib/hackdir** for the files listed below; this directory can be changed with the **–d** option. The **–s** option permits you to search the player record. Given the keyword **all**, **hack** lists all players; given the login name of a player, it lists all scores of that player.

FILES

record	top 100 list (start with an empty file)
news	changes or bugs (start with no news file)
data	information about objects and monsters
help	introductory information (no doubt outdated)
hh	compacted version of help
perm	empty file used for locking
rumors	texts for fortune cookies

NAME

hangman – computer version of the game hangman

SYNOPSIS

/usr/games/hangman

DESCRIPTION

In **hangman,** the computer picks a word from the on-line word list and you must try to guess it. The computer keeps track of which letters have been guessed and how many wrong guesses you have made on the screen in a graphic fashion.

FILES

/usr/dict/words	on-line word list

NAME

hunt – a multiplayer multiterminal game

SYNOPSIS

/usr/games/hunt[**–m**] [*hostname*] [**–l** *name*]

DESCRIPTION

The object of the game **hunt** is to kill off the other players. There are no rooms, no treasures, and no monsters. Instead, you wander around a maze, find grenades, trip mines, and shoot down walls and players.

Your score is the ratio of number of kills to number of times you entered the game and is only kept for the duration of a single session of **hunt**. The more players you kill before you die, the better your score is.

hunt normally looks for an active game on the local network; if none is found, it starts one up on the local host. One may specify the location of the game by giving the *hostname* argument.

hunt only works on crt (vdt) terminals with at least 24 lines, 80 columns, and cursor addressing. The screen is divided in to 3 areas. On the right hand side is the status area. It shows you how much damage you've sustained, how many charges you have left, who's in the game, who's scanning (the asterisk in front of the name), who's cloaked (the plus sign in front of the name), and other players' scores. Most of the rest of the screen is taken up by your map of the maze, except for the 24th line, which is used for longer messages that do not fit in the status area.

hunt uses the same keys to move as **vi** does, for instance, **h**,**j**,**k**, and **l** for left, down, up, right respectively. To change which direction you're facing in the maze, use the upper case version of the movement key (for instance, HJKL).

Other commands are:

f	Fire (in the direction you're facing) (Takes 1 charge)
g	Throw grenade (in the direction you're facing) (Takes 9 charges)
F	Throw satchel charge (Takes 25 charges)
G	Throw bomb (Takes 49 charges)
o	Throw small slime bomb (Takes 15 charges)
O	Throw big slime bomb (Takes 30 charges)
s	Scan (where other players are) (Takes 1 charge)
c	Cloak (where you are) (Takes 1 charge)
^L	Redraw screen
q	Quit

Knowing what the symbols on the screen often helps:

–|+	Walls
/\hl288u+288uDiagonal (deflecting) walls	
#	Doors (dispersion walls)
;	Small mine
g	Large mine
:	Shot
o	Grenade
O	Satchel charge
@	Bomb
s	Small slime bomb
$	Big slime bomb
><^v	You facing right, left, up, or down

} { i ! Other players facing right, left, up, or down

* Explosion

\ /
– * *E* – Grenade and large mine explosion
/ \

Satchel and bomb explosions are larger than grenades (5x5, 7x7, and 3x3 respectively).

Other helpful hints:

You can only fire in the direction you are facing.
You can only fire three shots in a row, then the gun must cool.
A shot only affects the square it hits.
Shots and grenades move 5 times faster than you do.
To stab someone,
    you must face that player and move at them.
Stabbing does 3 points worth of damage and shooting does 5 points.
You start with 15 charges and get 5 more for every new player.
A grenade affects the nine squares centered about the square it hits.
A satchel affects the twenty-five squares centered about the square it hits.
A bomb affects the forty-nine squares centered about the square it hits.
One small mine and one large mine is placed in the maze for every new player.
A mine has a 5% probability of tripping when you walk directly at it;
    50% when going sideways on to it; 95% when backing up on to it.
Tripping a mine costs you 5 points or 10 points respectively.
Defusing a mine is worth 1 charge or 9 charges respectively.
You cannot see behind you.
Scanning lasts for (20 times the number of players) turns.
    Scanning takes 1 ammo charge, so do not waste all your charges scanning.
You get 2 more damage capacity points and 2 damage points taken away
    whenever you kill someone.
Maximum typeahead is 5 characters.
A shot destroys normal (for instance, non-diagonal, non-door) walls.
Diagonal walls deflect shots and change orientation.
Doors disperse shots in random directions (up, down, left, right).
Diagonal walls and doors cannot be destroyed by direct shots but may
    be destroyed by an adjacent grenade explosion.
Walls regenerate, reappearing in the order they were destroyed.
    One percent of the regenerated walls will be diagonal walls or doors. When a wall is generated directly beneath a player, he is thrown in a random direction for a random period of time. When he lands, he sustains damage (up to 20 percent of the amount of damage he had before impact); that is, the less damage he had, the more nimble he is and therefore less likely to hurt himself on landing.

ENVIRONMENT

The environment variable **HUNT** is checked to get the player name. If you do not have this variable set, **hunt** will ask you what name you want to play under. You may also set up a single character keyboard map, but then you have to enumerate the options. For example:

setenv HUNT "name=Sneaky,mapkey=zoFfGg1f2g3F4G"

sets the player name to Sneaky, and the maps **z** to **o**, **F** to **f**, **G** to **g**, **1** to **f**, **2** to **g**, **3** to **F**, and **4** to **G**.

The *mapkey* option must be last.

It is a boring game if you are the only one playing.

OPTIONS

-m You enter the game as a monitor (you can see the action but you cannot play).

-l *name* Enter the game as player *name*.

FILES

/usr/games/lib/hunt.driver game coordinator

LIMITATIONS

hunt normally drives up the load average to be about (number_of_players + 0.5) greater than it would be without a **hunt** game executing. A limit of three players per host and nine players total is enforced by **hunt**.

BUGS

To keep up the pace, not everything is as realistic as possible.

NAME

life – John Conway's game of life

SYNOPSIS

/usr/games/life

AVAILABILITY

This game is available with the *Games* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

life is a program that plays John Conway's game of life. It only runs under **sunview**(1).

When invoked, **life** will display a window with a small control panel at the top, and a large drawing area at the bottom. You can create pieces in the drawing area with the left button, and erase them with the middle button. When you select **Run** in the control panel, the pieces will begin to evolve, and the drawing region will update itself at a speed controlled by the slider labeled with **Fast** and **Slow**. **life** keeps track of all the pieces even if they are not visible. The scroll bars surrounding the drawing region can be used to see pieces that have moved out of view. There are some standard patterns that can be drawn by popping up a menu in the drawing subwindow.

The meaning of the items in the first row of the control panel (from left to right) are as follows. If you click on the picture which looks like a tic-tac-toe board, a grid will appear in the drawing region. If you click on **Step**, the mode will change from run mode (where the pieces update continuously) to step mode (where an update is only done when you click on **Step**). Following **Gen** is a number indicating the number of generations that have occurred. The button marked **Find** will scroll so that at least one piece is in view. This is useful when all the pieces disappear from view. The button marked **Clear** will clear the drawing region, but leave the other controls unchanged. **Reset** will reset all the panel controls, but will not erase any of the pieces, and **Quit** Exits the tool. The second row contains two sliders. The first controls the update speed when in run mode, the second controls the size of the pieces.

SEE ALSO

sunview(1)

NAME

mille – play Mille Bornes

SYNOPSIS

/usr/games/mille [file]

DESCRIPTION

mille plays a two-handed game reminiscent of the Parker Brother's game of Mille Bornes with you. The rules are described below. If a file name is given on the command line, the game saved in that file is started.

When a game is started up, the bottom of the score window will contain a list of commands. They are:

P Pick a card from the deck. This card is placed in the 'P' slot in your hand.

D Discard a card from your hand. To indicate which card, type the number of the card in the hand (or "P" for the just-picked card) followed by a carriage-return or space. The carriage-return or space is required to allow recovery from typos which can be very expensive, like discarding safeties.

U Use a card. The card is again indicated by its number, followed by a carriage-return or space.

O Toggle ordering the hand. By default off, if turned on it will sort the cards in your hand appropriately. This is not recommended for the impatient on slow terminals.

Q Quit the game. This will ask for confirmation, just to be sure. Hitting DELETE (or RUBOUT) is equivalent.

S Save the game in a file. If the game was started from a file, you will be given an opportunity to save it on the same file. If you don't wish to, or you did not start from a file, you will be asked for the file name. If you type a RETURN character without a name, the save will be terminated and the game resumed.

R Redraw the screen from scratch. The command ^L (CTRL–L) will also work.

W Toggle window type. This switches the score window between the startup window (with all the command names) and the end-of-game window. Using the end-of-game window saves time by eliminating the switch at the end of the game to show the final score. Recommended for hackers and other miscreants.

If you make a mistake, an error message will be printed on the last line of the score window, and a bell will beep.

At the end of each hand or game, you will be asked if you wish to play another. If not, it will ask you if you want to save the game. If you do, and the save is unsuccessful, play will be resumed as if you had said you wanted to play another hand/game. This allows you to use the "S" command to reattempt the save. (The game itself is a product of Parker Brothers, Inc.)

SEE ALSO

curses(3V)

CARDS

Here is some useful information. The number in brackets after the card name is the number of that card in the deck:

Hazard	Repair	Safety
Out of Gas [2]	Gasoline [6]	Extra Tank [1]
Flat Tire [2]	Spare Tire [6]	Puncture Proof [1]
Accident [2]	Repairs [6]	Driving Ace [1]
Stop [4]	Go [14]	Right of Way [1]
Speed Limit [3]	End of Limit [6]	

25 – [10], 50 – [10], 75 – [10], 100 – [12], 200 – [4]

RULES

Object: The point of game is to get a total of 5000 points in several hands. Each hand is a race to put down exactly 700 miles before your opponent does. Beyond the points gained by putting down milestones, there are several other ways of making points.

Overview: The game is played with a deck of 101 cards. *Distance* cards represent a number of miles traveled. They come in denominations of 25, 50, 75, 100, and 200. When one is played, it adds that many miles to the player's trip so far this hand. *Hazard* cards are used to prevent your opponent from putting down Distance cards. With the exception of the *speed limit* card, they can only be played if your opponent has a *Go* card on top of the Battle pile. The cards are *Out of Gas*, *Accident*, *Flat Tire*, *Speed Limit*, and *Stop*. *Remedy* cards fix problems caused by Hazard cards played on you by your opponent. The cards are *Gasoline*, *Repairs*, *Spare Tire*, *End of Limit*, and *Go*. *Safety* cards prevent your opponent from putting specific Hazard cards on you in the first place. They are *Extra Tank*, *Driving Ace*, *Puncture Proof*, and *Right of Way*, and there are only one of each in the deck.

Board Layout: The board is split into several areas. From top to bottom, they are: **SAFETY AREA (unlabeled)**: This is where the safeties will be placed as they are played. **HAND**: These are the cards in your hand. **BATTLE**: This is the Battle pile. All the Hazard and Remedy Cards are played here, except the *Speed Limit* and *End of Limit* cards. Only the top card is displayed, as it is the only effective one. **SPEED**: The Speed pile. The *Speed Limit* and *End of Limit* cards are played here to control the speed at which the player is allowed to put down miles. **MILEAGE**: Miles are placed here. The total of the numbers shown here is the distance traveled so far.

Play: The first pick alternates between the two players. Each turn usually starts with a pick from the deck. The player then plays a card, or if this is not possible or desirable, discards one. Normally, a play or discard of a single card constitutes a turn. If the card played is a safety, however, the same player takes another turn immediately.

This repeats until one of the players reaches 700 points or the deck runs out. If someone reaches 700, they have the option of going for an *Extension*, which means that the play continues until someone reaches 1000 miles.

Hazard and Remedy Cards: Hazard Cards are played on your opponent's Battle and Speed piles. Remedy Cards are used for undoing the effects of your opponent's nastiness.

Go (Green Light) must be the top card on your Battle pile for you to play any mileage, unless you have played the *Right of Way* card (see below).

Stop is played on your opponent's *Go* card to prevent them from playing mileage until they play a *Go* card.

Speed Limit is played on your opponent's Speed pile. Until they play an *End of Limit* they can only play 25 or 50 mile cards, presuming their *Go* card allows them to do even that.

End of Limit is played on your Speed pile to nullify a *Speed Limit* played by your opponent.

Out of Gas is played on your opponent's *Go* card. They must then play a *Gasoline* card, and then a *Go* card before they can play any more mileage.

Flat Tire is played on your opponent's *Go* card. They must then play a *Spare Tire* card, and then a *Go* card before they can play any more mileage.

Accident is played on your opponent's *Go* card. They must then play a *Repairs* card, and then a *Go* card before they can play any more mileage.

Safety Cards: Safety cards prevent your opponent from playing the corresponding Hazard cards on you for the rest of the hand. It cancels an attack in progress, and *always entitles the player to an extra turn*.

Right of Way prevents your opponent from playing both *Stop* and *Speed Limit* cards on you. It also acts as a permanent *Go* card for the rest of the hand, so you can play mileage as long as there is not a Hazard card on top of your Battle pile. In this case only, your opponent can play Hazard cards directly on a Remedy card besides a Go card.

Extra Tank When played, your opponent cannot play an *Out of Gas* on your Battle Pile.

Puncture Proof When played, your opponent cannot play a *Flat Tire* on your Battle Pile.

Driving Ace When played, your opponent cannot play an *Accident* on your Battle Pile.

Distance Cards: Distance cards are played when you have a *Go* card on your Battle pile, or a Right of Way in your Safety area and are not stopped by a Hazard Card. They can be played in any combination that totals exactly 700 miles, except that *you cannot play more than two 200 mile cards in one hand*. A hand ends whenever one player gets exactly 700 miles or the deck runs out. In that case, play continues until neither someone reaches 700, or neither player can use any cards in their hand. If the trip is completed after the deck runs out, this is called *Delayed Action*.

Coup Fouré: This is a French fencing term for a counter-thrust move as part of a parry to an opponents attack. In Mille Bornes, it is used as follows: If an opponent plays a Hazard card, and you have the corresponding Safety in your hand, you play it immediately, even *before* you draw. This immediately removes the Hazard card from your Battle pile, and protects you from that card for the rest of the game. This gives you more points (see "Scoring" below).

Scoring: Scores are totaled at the end of each hand, whether or not anyone completed the trip. The terms used in the Score window have the following meanings:

Milestones Played: Each player scores as many miles as they played before the trip ended.

Each Safety: 100 points for each safety in the Safety area.

All 4 Safeties: 300 points if all four safeties are played.

Each Coup Fouré: 300 points for each Coup Fouré accomplished.

The following bonus scores can apply only to the winning player.

Trip Completed: 400 points bonus for completing the trip to 700 or 1000.

Safe Trip: 300 points bonus for completing the trip without using any 200 mile cards.

Delayed Action: 300 points bonus for finishing after the deck was exhausted.

Extension: 200 points bonus for completing a 1000 mile trip.

Shut-Out: 500 points bonus for completing the trip before your opponent played any mileage cards.

Running totals are also kept for the current score for each player for the hand (**Hand Total**), the game (**Overall Total**), and number of games won (**Games**).

NAME

monop – Monopoly game

SYNOPSIS

/usr/games/monop [*filename*]

DESCRIPTION

monop is reminiscent of the Parker Brother's game Monopoly, and monitors a game between 1 to 9 users. It is assumed that the rules of Monopoly are known. The game follows the standard rules, with the exception that, if a property would go up for auction and there are only two solvent players, no auction is held and the property remains unowned.

The game, in effect, lends the player money, so it is possible to buy something which you cannot afford. However, as soon as a person goes into debt, he must "fix the problem", that is, make himself solvent, before play can continue. If this is not possible, the player's property reverts to his debtee, either a player or the bank. A player can resign at any time to any person or the bank, which puts the property back on the board, unowned.

Any time that the response to a question is a *string*, for instance a name, place or person, you can type ? to get a list of valid answers. It is not possible to input a negative number, nor is it ever necessary.

USAGE

Commands

quit: Quit game. This allows you to quit the game. It asks you if you are sure.

print Print board. This prints out the current board. The columns have the following meanings (column headings are the same for the **where**, **own holdings**, and **holdings** commands):

Name	The first ten characters of the name of the square
Own	The *number* of the owner of the property.
Price	The cost of the property (if any)
Mg	This field has a '*' in it if the property is mortgaged
#	If the property is a Utility or Railroad, this is the number of such owned by the owner. If the property is land, this is the number of houses on it.
Rent	Current rent on the property. If it is not owned, there is no rent.

where: where players are: Tells you where all the players are. A '*' indicates the current player.

own holdings :
List your own holdings, that is, money, get-out-of-jail-free cards, and property.

holdings:
Holdings list. Look at anyone's holdings. It will ask you whose holdings you wish to look at. When you are finished, type **done**.

shell: Shell escape. Escape to a shell. When the shell dies, the program continues where you left off.

mortgage:
Mortgage property. Sets up a list of mortgageable property, and asks which you wish to mortgage.

unmortgage:
Unmortgage property. Unmortgage mortgaged property.

buy: Buy houses. Sets up a list of monopolies on which you can buy houses. If there is more than one, it asks you which you want to buy for. It then asks you how many for each piece of property, giving the current amount in parentheses after the property name. If you build in an unbalanced manner (a disparity of more than one house within the same monopoly), it asks you to re-input things.

sell: Sell houses. Sets up a list of monopolies from which you can sell houses. it operates in an analogous manner to **buy**

card: Card for jail. Use a get-out-of-jail-free card to get out of jail. If you are not in jail, or you do not have one, it tells you so.

pay: Pay for jail. Pay $50 to get out of jail, from whence you are put on Just Visiting. Difficult to do if you are not there.

trade: This allows you to trade with another player. It asks you whom you wish to trade with, and then asks you what each wishes to give up. You can get a summary at the end, and, in all cases, it asks for confirmation of the trade before doing it.

resign: Resign to another player or the bank. If you resign to the bank, all property reverts to its virgin state, and get-out-of-jail free cards revert to the deck.

save: Save game. Save the current game in a file for later play. You can continue play after saving, either by adding the file in which you saved the game after the **monop** command, or by using the **restore** command (see below). It will ask you which file you wish to save it in, and, if the file exists, confirm that you wish to overwrite it.

restore:
Restore game. Read in a previously saved game from a file. It leaves the file intact.

roll: Roll the dice and move forward to your new location. If you simply hit the RETURN key instead of a command, it is the same as typing *roll*.

FILES

/usr/games/lib/cards.pck chance and community chest cards

BUGS

No command can be given an argument instead of a response to a query.

NAME

moo – guessing game

SYNOPSIS

/usr/games/moo

DESCRIPTION

moo is a guessing game imported from England. The computer picks a number consisting of four distinct decimal digits. The player guesses four distinct digits being scored on each guess. A "cow" is a correct digit in an incorrect position. A "bull" is a correct digit in a correct position. The game continues until the player guesses the number (a score of four bulls).

NAME

number – convert Arabic numerals to English

SYNOPSIS

/usr/games/number

DESCRIPTION

number copies the standard input to the standard output, changing each decimal number to a fully spelled out version.

NAME

play – play audio files

SYNOPSIS

play [**–i**] [**–V**] [**–d** *dev*] [**–v** *vol*] [*filename* ...]

AVAILABILITY

This command is only available with the *Demos* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

play copies the named audio files to the audio device. Audio files named on the command line are played sequentially. If no filenames are present, the standard input stream is played. The special filename '–' may be used to read the standard input stream instead of a file.

The input files (including the standard input) must contain a valid audio file header. The encoding information in this header is matched against the capabilities of the audio device and, if the data formats are incompatible, an error message is printed and the file is skipped.

Minor deviations in sampling frequency (those less than 1%) are ordinarily ignored. This allows, for instance, data sampled at 8012 Hz to be played on an audio device that only supports 8000 Hz. If the **–V** option is specified, such deviations are flagged with warning messages.

OPTIONS

–i Print an error message and exit immediately if the audio device is unavailable (that is, another process currently has write access). **play** will ordinarily wait until it can obtain access to the device.

–V Verbose. Print messages to the standard error while waiting for access to the audio device or when sample rate deviations are detected.

–d *dev* Specify an alternate audio device to which output should be directed. If the **–d** option is not specified, **/dev/audio** is the default audio device.

–v *vol* Set the output volume to *vol* before playing begins. *vol* is an integer value between 0 and 100, inclusive. If this argument is not specified, the output volume remains at the level most recently set by any process.

–? Help. Print a command line usage message.

SEE ALSO

record(6)

WARNINGS

This program is furnished on an *as is* basis as a demonstration of audio applications programming.

NAME

primes – print all primes larger than some given number

SYNOPSIS

/usr/games/primes [*number*]

DESCRIPTION

primes reads a number from the standard input and prints all primes larger than the given number. If *number* is given as an argument, it uses that number rather than reading one from the standard input.

BUGS

It obviously cannot print *all* primes larger than some given number. It will not behave very sensibly when it overflows an **int**.

NAME

quiz – test your knowledge

SYNOPSIS

/usr/games/quiz [–i*filename*] [–**t**] [*category1 category2*]

DESCRIPTION

quiz gives associative knowledge tests on various subjects. It asks items chosen from *category1* and expects answers from *category2*. If no categories are specified, **quiz** gives instructions and lists the available categories.

quiz tells a correct answer whenever you type a bare newline. At the end of input, upon interrupt, or when questions run out, **quiz** reports a score and terminates.

The –**t** flag specifies 'tutorial' mode, where missed questions are repeated later, and material is gradually introduced as you learn.

The –**i** flag causes the named file to be substituted for the default index file. The lines of these files have the syntax:

```
line     = category newline | category ':' line
category = alternate | category 'l' alternate
alternate = empty | alternate primary
primary  = character | '[' category ']' | option
option   = '{' category '}'
```

The first category on each line of an index file names an information file. The remaining categories specify the order and contents of the data in each line of the information file. Information files have the same syntax. Backslash '\' is used as with **sh**(1) to quote syntactically significant characters or to insert transparent newlines into a line. When either a question or its answer is empty, **quiz** will refrain from asking it.

FILES

/usr/games/quiz.k/*

BUGS

The construct 'a|ab' doesn't work in an information file. Use 'a{b}'.

NAME

rain – animated raindrops display

SYNOPSIS

/usr/games/rain

DESCRIPTION

rain's display is modeled after the VAX/VMS program of the same name. The terminal has to be set for 9600 baud to obtain the proper effect.

As with all programs that use **termcap**, the TERM environment variable must be set (and exported) to the type of the terminal being used.

FILES

/etc/termcap

NAME

random – select lines randomly from a file

SYNOPSIS

/usr/games/random [**–er**] [*divisor*]

DESCRIPTION

random acts as a text filter, randomly selecting lines from its standard input to write to the standard output. The probability that a given line is selected is normally 1/2; if a *divisor* is specified, it is treated as a floating-point number, and the probability is 1/*divisor* instead.

OPTIONS

–e Don't read the standard input or write to the standard output. Instead, exit with a random exit status between 0 and 1, or between 0 and *divisor*-1 if *divisor* is specified.

–r Don't buffer the output. If **–r** is not used, output is buffered in blocks, or line-buffered if the standard output is a terminal.

NAME

raw2audio – convert raw audio data to audio file format

SYNOPSIS

raw2audio [**–f**] [**–c** *chan*] [**–e** *enc*] [**–i** *info*] [**–o** *cnt*] [**–p** *bits*] [**–s** *rate*] [*filename* ...]

AVAILABILITY

This command is only available with the *Demos* installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

raw2audio adds an audio file header to the named raw data files. The encoding information in this header is taken from the command line options.

If no filenames are specified, **raw2audio** reads raw data from the standard input stream and writes an audio file to the standard output. If a target file is a symbolic link, the underlying file will be rewritten.

OPTIONS

–f Force. If an input file already contains an audio file header, **raw2audio** ordinarily prints a warning message and skips the file. If the **–f** flag is specified, the old file header, including the 'information' field, is replaced.

–c *chan* Specify the number of interleaved audio channels in each sample frame. If not specified, a single channel is assumed.

–e *enc* Specify the encoding type. *enc* may be one of the following: **ULAW**, **LINEAR**, or **FLOAT**, corresponding to μ-law, integer **PCM**, and IEEE floating-point formats, respectively. If not specified, μ-law encoding is assumed.

–i *info* Specify the 'information' field of the output file header.

–o *cnt* Specify the number of bytes to skip in the audio data stream. This option may be used, for instance, to extract audio data from files containing unrecognizable file headers.

–s *rate* Specify the sample rate frequency, in Hz. If not specified, the sample rate defaults to 8000 Hz.

–p *bits* Specify the sound unit size, in bits. If not specified, the precision defaults to 8 bits.

–? Help. Print a command line usage message.

SEE ALSO

play(6), **record**(6)

WARNINGS

This program is furnished on an *as is* basis as a demonstration of audio applications programming.

NAME

record – record an audio file

SYNOPSIS

record [**–a**] [**–f**] [**–d** *dev*] [**–i** *info*] [**–t** *time*] [**–v** *vol*] [*filename*]

AVAILABILITY

This command is only available with the *Demos* installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

record copies audio data from the audio device to a named audio file. The output file will be prefixed by an audio file header. The encoding information in this header is taken from the configuration of the audio device. If no filename is present, or if the special filename '–' is specified, output is directed to the standard output stream.

Recording begins immediately and continues until a SIGINT signal (CTRL-C) is received. If the **–t** option is specified, *record* stops when the specified quantity of data has been recorded.

If the audio device is unavailable (that is, another process currently has read access), **record** prints an error message and exits immediately.

OPTIONS

–a Append the data on the end of the named audio file. The audio encoding of the file must match the audio device configuration.

–f Force. When the **–a** flag is specified, the sample rate of the audio file must match the device configuration. If the **–f** flag is also specified, sample rate differences are ignored, with a warning message printed to the standard error.

–d *dev* Specify an alternate audio device from which input should be taken. If the **–d** option is not specified, **/dev/audio** is used as the default audio device.

–i *info* The 'information' field of the output file header is set to the string specified by the *info* argument. This option may not be specified in conjunction with the **–a** argument.

–t *time* The *time* argument specifies the maximum length of time to record. Time may be specified as a floating-point value, indicating the number of seconds, or in the form: *hh*:*mm*:*ss*.**dd**, where hour and minute specifications are optional.

–v *vol* Specify the recording gain. *vol* is an integer value between 0 and 100, inclusive. If this argument is not specified, the input volume will remain at the level most recently set by any process.

–? *Help*: Print a command line usage message.

SEE ALSO

play(6)

WARNINGS

This program is furnished on an *as is* basis as a demonstration of audio applications programming.

NAME

robots – fight off villainous robots

SYNOPSIS

/usr/games/robots [**–sjta**] [*scorefile*]

DESCRIPTION

robots pits you against evil robots, who are trying to kill you (which is why they are evil). Fortunately for you, even though they are evil, they are not very bright and have a habit of bumping into each other, thus destroying themselves. In order to survive, you must get them to kill each other off, since you have no offensive weaponry.

Since you are stuck without offensive weaponry, you are endowed with one piece of defensive weaponry: a teleportation device. When two robots run into each other or a junk pile, they die. If a robot runs into you, you die. When a robot dies, you get 10 points, and when all the robots die, you start on the next field. This keeps up until they finally get you.

Robots are represented on the screen by a '+', the junk heaps from their collisions by a '*', and you (the good guy) by a '@'.

The commands are:

h move one square left

l move one square right

k move one square up

j move one square down

y move one square up and left

u move one square up and right

b move one square down and left

n move one square down and right

. (also space) do nothing for one turn

HJKLBNYU
run as far as possible in the given direction

> do nothing for as long as possible

t teleport to a random location

w wait until you die or they all do

q quit

^L redraw the screen

All commands can be preceded by a count.

If you use the '**w**' command and survive to the next level, you will get a bonus of 10% for each robot which died after you decided to wait. If you die, however, you get nothing. For all other commands, the program will save you from typos by stopping short of being eaten. However, with '**w**' you take the risk of dying by miscalculation.

Only five scores are allowed per user on the score file. If you make it into the score file, you will be shown the list at the end of the game. If an alternate score file is specified, that will be used instead of the standard file for scores.

OPTIONS

–s Do not play, just show the score file.

–j Jump, when you run, don't show any intermediate positions; only show things at the end. This is useful on slow terminals.

–t Teleport automatically when you have no other option. This is a little disconcerting until you get used to it, and then it is very nice.

–a Advance into the higher levels directly, skipping the lower, easier levels.

FILES

/usr/games/lib/robots_roll the score file

BUGS

Bugs? You *crazy*, man?!?

NAME

rotcvph – rotate convex polyhedron

SYNOPSIS

/usr/demo/rotcvph*filename*

DESCRIPTION

rotcvph rotates a convex polyhedron with hidden surfaces removed. Using the SunCore Graphics Package, a 3-D projection is drawn on the Sun Monochrome Bitmap Display. The mandatory file argument contains a polygonal object definition as described below.

Initially the program displays a fixed view of the object. The following commands may be typed at any time:

n Display successive views with no waiting.

w Wait for SPACE to be typed before displaying each view.

q Exit the program.

The format of the polygonal object definition is illustrated by this example of the definition of a pyramid:

```
        5       5
-1.0 1.0 -1.0 1.0 -1.0 1.0
1.0  1.0 -1.0
1.0 -1.0 -1.0
-1.0 -1.0 -1.0
-1.0  1.0 -1.0
0.0  0.0  1.0
4       4 3 2 1
3       1 5 4
3       2 5 1
3       3 5 2
3       4 5 3
```

The first line gives the number of vertices followed by the number of polygons. The second line gives the coordinates of a bounding box for the object. Minimum and maximum coordinate values are given for each of three dimensions in the order *minx*, *maxx*, *miny*, *maxy*, *minz*, *maxz*. Lines 3 through v+2 (where v is the number of vertices) give vertex coordinates in the order *x*, *y*, ,IR z . Lines v+3 through v+p+2 (where p is the number of polygons) give polygon descriptions. The first number is the number of vertices for the polygon. Succeeding numbers on the line are indices into the vertex list. Polygons should be planar. Coordinates are given in floating point format and everything else is integer. Entries on a given line are separated by arbitrary whitespace. A maximum of 400 vertices and 400 polygons may be defined. The polygon definitions may contain a maximum of 1600 instances of the vertices. **/usr/demo/data** contains several object definition files, including **icosa.dat**, **socbal.dat**, and **pyramid.dat**.

The above format may be used to define non-convex objects. The program will display these objects but hidden surface computations will not be done correctly.

FILES

/usr/demo/data/*.dat	sample object definition files
icosa.dat	
socbal.dat	
pyramid.dat	

BUGS

All floating point transformations are done twice for each view, once to draw the object and once to undraw it.

Lines which are common to two visible polygons in a view are drawn twice, once for each polygon.

NAME

snake, snscore – display chase game

SYNOPSIS

/usr/games/snake [**–w***n*] [**–l***n*]
/usr/games/snscore

DESCRIPTION

snake is a display-based game which must be played on a CRT terminal from among those supported by **vi**(1). The object of the game is to make as much money as possible without getting eaten by the snake. The **–l** and **–w** options allow you to specify the length and width of the field. By default the entire screen (except for the last column) is used.

You are represented on the screen by an I. The snake is 6 squares long and is represented by S's. The money is $, and an exit is #. Your score is posted in the upper left hand corner.

You can move around using the same conventions as **vi**(1), the h, j, k, and l keys work, as do the arrow keys. Other possibilities include:

sefc These keys are like hjkl but form a directed pad around the d key.

HJKL These keys move you all the way in the indicated direction to the same row or column as the money. This does *not* let you jump away from the snake, but rather saves you from having to type a key repeatedly. The snake still gets all his turns.

SEFC Likewise for the upper case versions on the left.

ATPB These keys move you to the four edges of the screen. Their position on the keyboard is the mnemonic, for example, P is at the far right of the keyboard.

x This lets you quit the game at any time.

p Points in a direction you might want to go.

w Space warp to get out of tight squeezes, at a price.

! Shell escape

^Z Suspend the snake game, on systems which support it. Otherwise an interactive shell is started up.

To earn money, move to the same square the money is on. A new $ will appear when you earn the current one. As you get richer, the snake gets hungrier. To leave the game, move to the exit (#).

A record is kept of the personal best score of each player. Scores are only counted if you leave at the exit, getting eaten by the snake is worth nothing.

As in pinball, matching the last digit of your score to the number which appears after the game is worth a bonus.

To see who wastes time playing snake, run **/usr/games/snscore.**

FILES

/usr/games/lib/snakerawscores	database of personal bests
/usr/games/lib/snake.log	log of games played

BUGS

When playing on a small screen, it's hard to tell when you hit the edge of the screen.

The scoring function takes into account the size of the screen. A perfect function to do this equitably has not been devised.

NAME

soundtool – audio play/record tool

SYNOPSIS

soundtool

AVAILABILITY

This command is only available with the *Demos* installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

soundtool is a SunView demonstration program that allows recording, playing, and simple editing of audio data. The display consists of six regions: a play/record control panel, a function control panel, an oscilloscope, a display control panel, a waveform display panel, and a pop-up audio status panel.

Play/Record Control Panel

Play/Stop

Clicking this button plays the currently selected region of data. While data is playing this button becomes a **Stop** button. If audio output is busy when **Play** is started, this button displays **Waiting**. When the device is available, the button switches to **Stop** and audio output begins. Clicking on the **Waiting** button resets the tool to the idle state.

Record/Stop

Clicking this button starts the recording of data from the audio input port that is wired to the 8-pin mini-DIN connector on the back of SPARCstation 1 systems. While recording is in progress, this button becomes a **Stop** button. If audio input is busy when **Record** is selected, an alert pops up and the tool resets to the idle state. A maximum of 5 minutes may be recorded at a time.

Pause Clicking this button while playing or recording suspends the current operation. The button becomes a **Resume** button that may be selected to continue the suspended operation.

Describe

Clicking this button brings up the "Audio Status Panel". If the panel was already visible, clicking this button removes it.

Quit Clicking this button exits **soundtool**.

Play Volume

This slider adjusts the playback volume. Volume levels between 0 and 100 may be selected, where 0 represents infinite attenuation and 100 is maximum gain.

Record Volume

This slider adjusts the recording level in the range 0 to 100.

Output To

This selector switches the audio output port between the built-in speaker and the external headphone jack.

Looping

When **Looping** is disabled, the current data region (that is, the data between the two markers in the waveform display) is played once. If **Looping** is enabled, the selected data plays endlessly until the **Stop** button is pressed.

Function Control Panel

Load Clicking **Load** reads in the audio file specified by the **Directory** and **File** fields. If the named file does not contain a valid audio header, the raw data is copied into the buffer and an alert is displayed. Clicking the **Store** button at that point rewrites the file with the proper audio file header.

Arbitrarily large audio files may be loaded. However, system swap space resources may be depleted (one minute of SPARCstation 1 audio data consumes roughly .5 Mbyte of swap space).

Store Clicking **Store** writes the selected data region into the file specified by the **Directory** and **File** fields. If the named file exists, an alert will request confirmation of the operation.

Append

Clicking **Append** appends the selected data region to the file specified by the **Directory** and **File** fields. The named file must contain a valid audio file header.

Directory

The **Directory** field specifies a directory path in which to look for audio files.

File The **File** field designates the file to be loaded from, stored to, or appended to. Holding down the right mouse button on this field presents a menu of audio files in the currently designated directory. All files that contain a valid audio file header, or whose names have the suffix **.au** or **.snd**, are listed.

Oscilloscope

When the program is in the idle state and the cursor is in the waveform display panel, the oscilloscope acts as a magnifying glass, displaying the region of the audio waveform that is currently under the cursor. When the program is playing or recording, the oscilloscope displays the data that is currently being transferred. Note: there is a small time lag in the display of recorded data, due to the fact that the audio device driver buffers input data and delivers it to the application in discrete segments.

Display Control Panel

Zoom The **Zoom** slider adjusts the compression factor used in the display of the waveform. The upper compression limit is chosen so that the entire waveform fits in the waveform display panel. The lower limit is restricted by the ability to manipulate large scrolling regions in SunView. Adjustment of the **Zoom** slider ordinarily results in data compression or expansion around the center of the currently displayed waveform. If the waveform display contains one or both data selection markers, an attempt is made to keep at least a portion of the selected data region in the window.

The magnified waveform presented in the oscilloscope display is unaffected by the **Zoom** value. However, cursor movement over the waveform reflects the current compression; that is, lower **Zoom** values result in finer granularity of mouse movement.

Waveform Display Panel

The waveform display shows all or part of the current waveform, depending on the current **Zoom** value. Scrolling of the waveform may be achieved either by using the scrollbar or by dragging the waveform to the right or left while holding the middle mouse button down. Note: scrolling is disabled when the entire waveform is being displayed (that is, when the **Zoom** value is at its maximum).

In some cases, it is desirable to identify a subset of the waveform. For instance, the **Play**, **Store**, and **Append** functions operate on a selected region, rather than the entire waveform. The currently selected region of interest is delimited by dashed vertical lines. A new region may be selected by clicking the left or right mouse button and dragging it across the desired region of interest. Alternatively, a single click on the left or right mouse buttons adjusts the start or end points.

Audio Status Panel

This panel is displayed (or removed) when the **Describe** button is pressed. It contains fields that describe the data in the buffer.

Sample Rate

This field displays the sampling frequency, in samples per second.

Channels

This field denotes the number of interleaved channels of audio data.

Precision

This field identifies the encoding precision, in bits per sample.

Encoding
This field displays the encoding format.

Total Length
This field shows the length of the entire data buffer, in the form *hh:mm:ss.dd*.

Selection
This field identifies the start and end times of the currently selected region of interest.

Info String
When an audio file is loaded, the first 80 characters of the information field of the audio header are displayed in this field. This string may be edited, though the new information is only written out when the **Store** operation is performed.

BUGS

Currently, **soundtool** is capable of displaying only 8-bit μ-law encoded data. This restriction should be removed.

Audio files should be mapped in order to reduce the swap space requirements. The limit on recording length should also be removed.

SunView scrollbars operate on canvases whose virtual size is given by a short integer (that is, 16 bits). This ridiculous constraint is the reason for the lower limit on zooming. Because of this, the accuracy of start and end point selection is reduced when the data buffer is large.

Region selections made over the waveform display panel work best when the click and drag paradigm is used. Adjusting the start or end points by a single click is susceptible to error; that is, if the mouse moves slightly between the button down and up events, the result is a very small selection.

SEE ALSO

gaintool(6), **play**(6), **raw2audio**(6), **record**(6)

WARNINGS

This program is furnished on an *as is* basis as a demonstration of audio applications programming.

NAME

suncoredemos – demonstrate SunCore Graphics Package

SYNOPSIS

/usr/demo/cproduct
/usr/demo/cshademo

AVAILABILITY

This command is only available with the *Demos* installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

suncoredemos is a collection of simple programs demonstrating the SunCore Graphics Package. Each program is briefly described below. These programs generate all graphics output using subroutine calls to SunCore. To exit each program, generate an interrupt signal by typing the appropriate key (usually DELETE).

cproduct Color Sun architecture demo (requires Sun Color Graphics Display).

cshademo Shaded surface polygons demo (requires Sun Color Graphics Display).

NAME

sunview_demos, canvas_demo, cursor_demo – Window-System demonstration programs

SYNOPSIS

/usr/demo/canvas_demo

/usr/demo/cursor_demo

AVAILABILITY

These demos are available with the *SunView Demos* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

canvas_Demo

canvas_demo demonstrates the capabilities of the canvas subwindow package. It consists of two subwindows: a control panel and a canvas. By adjusting the items on the control panel, you can manipulate the attributes of the canvas, and see the results.

cursor_Demo

cursor_demo demonstrates what you can do with cursors. A single control panel is provide for adjusting the various cursor attributes. As you adjust the items on the control panel, the panel's cursor changes in appearance.

NAME

trek – trekkie game

SYNOPSIS

/usr/games/trek [[**–a**] *filename*]

DESCRIPTION

trek is a game of space glory and war. Below is a summary of commands. For complete documentation, see *Trek* by Eric Allman.

If a filename is given, a log of the game is written onto that file. If the **–a** flag is given before the filename, that file is appended to, not truncated.

The game will ask you what length game you would like. Valid responses are "short", "medium", and "long". You may also type "restart", which restarts a previously saved game. You will then be prompted for the skill, to which you must respond "novice", "fair", "good", "expert", "commodore", or "impossible". You should normally start out with a novice and work up.

In general, throughout the game, if you forget what is appropriate the game will tell you what it expects if you just type in a question mark.

COMMAND SUMMARY

abandon
cloak **u**p/**d**own
computer request; ...
destruct
help
lrscan
phasers **a**utomatic amount
phasers **m**anual amt1 course1 spread1 ...
torpedo course [yes] angle/**n**o
ram course distance
shell
srscan [yes/**n**o]
status
undock
warp warp_factor

capture

damages
dock
impulse course distance
move course distance

rest time
shields **up**/**d**own

terminate yes/no
visual course

NAME

vwcvph – view convex polyhedron

SYNOPSIS

/usr/demo/vwcvph *filename*

DESCRIPTION

vwcvph allows the user to view a convex polyhedron from various positions with hidden surfaces removed. The viewing position is selected using the mouse. Using the SunCore Graphics Package, a 3-D projection is drawn on the Sun Monochrome Bitmap Display. The mandatory file argument contains a polygonal object definition as described in the manual page for **/usr/demo/rotcvph.**

The program operates in two modes: **DisplayObject** mode and **SelectView** mode. The program starts in **DisplayObject mode:**

DisplayObject:

The object is displayed in 3-D perspective with hidden surfaces removed. Type **q** while in this mode to exit the program. Press RIGHT mouse button to switch to **SelectView** mode.

SelectView:

Schematic projections of the outline of the object are shown and the mouse is used to select a viewing position. Press LEFT mouse button to set *x* and MIDDLE mouse button to set *y* in the *Front View*. Use MIDDLE mouse button to set *z* in the *Top View*. Press RIGHT mouse button to switch to **DisplayObject** mode.

The view shown in **DisplayObject** mode is drawn using the conventions that the viewer is always looking from the viewing position toward the center of the object and that the positive *y* axis on the screen is the projection of the positive *y* axis in object coordinates.

The input file may define non-convex objects. The program will display these objects but hidden surface computations will not be done correctly.

FILES

/usr/demo/data/*.dat	sample object definition files

BUGS

Lines which are common to two visible polygons in a view are drawn twice, once for each polygon.

NAME

worm – play the growing worm game

SYNOPSIS

/usr/games/worm [*size*]

DESCRIPTION

In **worm,** you are a little worm, your body is the o **'s** on the screen and your head is the @ . You move with the hjkl keys (as in the game **snake**). If you don't press any keys, you continue in the direction you last moved. The upper case HJKL keys move you as if you had pressed several (9 for HL and 5 for JK) of the corresponding lower case key (unless you run into a digit, then it stops).

On the screen you will see a digit; if your worm eats the digit it will grow longer, the actual amount longer depends on which digit it was that you ate. The object of the game is to see how long you can make the worm grow.

The game ends when the worm runs into either the sides of the screen, or itself. The current score (how much the worm has grown) is kept in the upper left corner of the screen.

The optional argument, if present, is the initial length of the worm.

BUGS

If the initial length of the worm is set to less than one or more than 75, various strange things happen.

NAME

worms – animate worms on a display terminal

SYNOPSIS

/usr/games/worms [**–field**] [**–length #**] [**–number #**] [**–trail**]

DESCRIPTION

–field makes a "field" for the worm(s) to eat; **–trail** causes each worm to leave a trail behind it. You can figure out the rest by yourself.

FILES

/etc/termcap

SEE ALSO

Snails by Karl Heuer

BUGS

The lower-right-hand character position will not be updated properly on a terminal that wraps at the right margin.

Terminal initialization is not performed.

NAME

wump – the game of hunt the wumpus

SYNOPSIS

/usr/games/wump

DESCRIPTION

wump plays the game of 'Hunt the Wumpus.' A Wumpus is a creature that lives in a cave with several rooms connected by tunnels. You wander among the rooms, trying to shoot the Wumpus with an arrow, meanwhile avoiding being eaten by the Wumpus and falling into Bottomless Pits. There are also Super Bats which are likely to pick you up and drop you in some random room.

The program asks various questions which you answer one per line; it will give a more detailed description if you want.

This program is based on one described in *People's Computer Company*, 2, 2 (November 1973).

NAME

intro – miscellaneous useful information pages

DESCRIPTION

This section contains miscellaneous documentation, mostly in the area of text processing macro packages for **troff**(1).

A 7V section number means one or more of the following:

- The man page documents System V behavior only.
- The man page documents default SunOS behavior, and System V behavior as it differs from the default behavior. These System V differences are presented under **SYSTEM V** section headers.
- The man page documents behavior compliant with *IEEE Std 1003.1-1988* (POSIX.1).

LIST OF MISC. TABLES

Name	Appears on Page	Description
ansic	**ansic**(7V)	ANSI C (draft of December 7 1988) lint library
ascii	**ascii**(7)	map of ASCII character set
bsd	**bsd**(7)	overview of the Berkeley 4.3 environment
eqnchar	**eqnchar**(7)	special character definitions for eqn
filesystem	**filesystem**(7)	file system organization
hier	**hier**(7)	file system hierarchy
iso_8859_1	**iso_8859_1**(7)	map of character set
man	**man**(7)	macros to format Reference Manual pages
me	**me**(7)	macros for formatting papers
ms	**ms**(7)	text formatting macros
posix	**posix**(7V)	overview of the IEEE Std 1003.1-1988 (POSIX.1) environment
SunOS	**sunos**(7)	overview of the SunOS Release 4.1 environment
svidii	**svidii**(7V)	overview of the System V environment
svidiii	**svidiii**(7V)	SVIDIII lint library
xopen	**x/open**(7V)	overview of the XPG Issue 2 (X/Open) environment

NAME

ansic – ANSI C (draft of December 7 1988) lint library

SYNOPSIS

/usr/5bin/lint –n –lansic *ansic_src.c*

AVAILABILITY

This environment is not available under SunOS Release 4.1. The environment that most closely approximates an ANSI C environment is the System V environment. The System V environment is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

ANSI C is a proposed standard for the C language. SunOS Release 4.1 does not currently fully support ANSI C applications. It does support many of the functions described by the ANSI C draft. This man page does not imply that the functions supported by SunOS Release 4.1 and the functions described by the ANSI C draft perform identically. The ANSI C lint library is intended solely as a porting aid.

The ANSI C lint library consists exclusively of ANSI C functions. Users may lint their code with the **–n –lansic** options to catch all non-ANSI C features.

Certain functions defined in the ANSI C lint library are not available in the C library but are available. In particular, math functions are made available only when the **–lm** option is added to **cc**(1V) or **ld**(1) commands.

Other ANSI C functions not supported at all in SunOS Release 4.1 are **raise( )**, **fgetpos( )**, **fsetpos( )**, **div( )**, **ldiv( )**, **strtoul( )**, **strerror( )**, and **difftime( )**.

FILES

/usr/5lib/lint/llib-lansic*

ANSI C lint library

SEE ALSO

lint(1V), **bsd**(7), **posix**(7V), **sunos**(7), **svidii**(7V), **svidiii**(7V), **xopen**(7V)

NAME

ascii – map of ASCII character set

SYNOPSIS

cat /usr/pub/ascii

DESCRIPTION

/usr/pub/ascii is a map of the ASCII character set, to be printed as needed. It contains octal and hexadecimal values for each character. While not included in that file, a chart of decimal values is also shown here.

Octal — Character

000	NUL	001	SOH	002	STX	003	ETX	004	EOT	005	ENQ	006	ACK	007	BEL
010	BS	011	HT	012	NL	013	VT	014	NP	015	CR	016	SO	017	SI
020	DLE	021	DC1	022	DC2	023	DC3	024	DC4	025	NAK	026	SYN	027	ETB
030	CAN	031	EM	032	SUB	033	ESC	034	FS	035	GS	036	RS	037	US
040	SP	041	!	042	"	043	#	044	$	045	%	046	&	047	´
050	(	051	)	052	*	053	+	054	,	055	–	056	.	057	/
060	0	061	1	062	2	063	3	064	4	065	5	066	6	067	7
070	8	071	9	072	:	073	;	074	<	075	=	076	>	077	?
100	@	101	A	102	B	103	C	104	D	105	E	106	F	107	G
110	H	111	I	112	J	113	K	114	L	115	M	116	N	117	O
120	P	121	Q	122	R	123	S	124	T	125	U	126	V	127	W
130	X	131	Y	132	Z	133	[	134	\	135	]	136	^	137	_
140	`	141	a	142	b	143	c	144	d	145	e	146	f	147	g
150	h	151	i	152	j	153	k	154	l	155	m	156	n	157	o
160	p	161	q	162	r	163	s	164	t	165	u	166	v	167	w
170	x	171	y	172	z	173	{	174	|	175	}	176	~	177	DEL

Hexadecimal — Character

00	NUL	01	SOH	02	STX	03	ETX	04	EOT	05	ENQ	06	ACK	07	BEL
08	BS	09	HT	0A	NL	0B	VT	0C	NP	0D	CR	0E	SO	0F	SI
10	DLE	11	DC1	12	DC2	13	DC3	14	DC4	15	NAK	16	SYN	17	ETB
18	CAN	19	EM	1A	SUB	1B	ESC	1C	FS	1D	GS	1E	RS	1F	US
20	SP	21	!	22	"	23	#	24	$	25	%	26	&	27	´
28	(	29	)	2A	*	2B	+	2C	,	2D	–	2E	.	2F	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3A	:	3B	;	3C	<	3D	=	3E	>	3F	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4A	J	4B	K	4C	L	4D	M	4E	N	4F	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5A	Z	5B	[	5C	\	5D	]	5E	^	5F	_
60	`	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6A	j	6B	k	6C	l	6D	m	6E	n	6F	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7A	z	7B	{	7C	|	7D	}	7E	~	7F	DEL

Decimal — Character

0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK	7	BEL
8	BS	9	HT	10	NL	11	VT	12	NP	13	CR	14	SO	15	SI
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN	23	ETB
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS	31	US
32	SP	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	DEL

FILES

/usr/pub/ascii Online chart of octal and hexadecimal values for the ASCII character set.

NAME

bsd – overview of the Berkeley 4.3 environment

SYNOPSIS

/usr/bin/lint –n –lbsd *bsd_src.c*

DESCRIPTION

BSD 4.3 is a set of functions and headers. The SunOS Release 4.1 is a superset of BSD 4.3. It includes all of the functionality described in the BSD 4.3 documentation. See **sunos**(7) for an overview of SunOS functionality.

Note: there may be some cases where the coexistence of another environment overrides the BSD 4.3 semantics. In particular, when there has been a point of conflict between POSIX.1 and BSD 4.3, POSIX.1 has won (see **setsid**(8V) for such an example).

Many man pages are marked with a "V" after the section number, indicating some sort of System V conformance. BSD 4.3 functions are also documented on these man pages, as well as on man pages without the "V" section suffix.

By default, the user will get a superset of the BSD 4.3 environment. No path modifications should be necessary. The typical path is **set path = (/usr/ucb /bin /usr/bin)**

LINT

As a portability aid, Sun is providing a lint library that consists exclusively of BSD 4.3 functions. Users may lint their code with the **–n –lbsd** options to catch all non-BSD 4.3 features.

BSD, as with most other environments, continues to evolve. The **–lbsd** lint library will always refer to the most recent BSD release supported by Sun. Some applications may wish to port to a particular release of BSD. They may safely use the more specific name of **–l4.3bsd** (currently the same as **–lbsd**). Lint libraries for BSD releases earlier than 4.3 are not currently available. 4.3 BSD is sufficiently close to 4.2 BSD that the 4.3 BSD lint library usually works.

FILES

/usr/bin/*	BSD 4.3 and SunOS specific executables
/usr/ucb/*	BSD 4.3 derived executables
/usr/include/*	BSD 4.3 and SunOS specific header files
/usr/lib/*	BSD 4.3 and SunOS specific library files
/usr/lib/lint/llib-lbsd*	BSD 4.3 lint library

SEE ALSO

lint(1V), **ansic**(7V), **posix**(7V), **sunos**(7), **svidii**(7V), **svidiii**(7V), **xopen**(7V), **setsid**(8V)

NAME

eqnchar – special character definitions for eqn

SYNOPSIS

eqn /usr/pub/eqnchar [*filename*] | **troff** [*options*]

neqn /usr/pub/eqnchar [*filename*] | **nroff** [*options*]

DESCRIPTION

eqnchar contains **troff**(1) and **nroff**(1) character definitions for constructing characters that are not available on the Graphic Systems typesetter. These definitions are primarily intended for use with **eqn**(1) and **eqn**(1). It contains definitions for the following characters

ciplus	⊕	*//*	//	*square*	□
citimes	⊗	*langle*	〈	*circle*	○
wig	~	*rangle*	〉	*blot*	□
-wig	≈	*hbar*	ℏ	*bullet*	•
>wig	≳	*ppd*	⊥	*prop*	∝
<wig	≲	*<->*	↔	*empty*	∅
=wig	≅	*<=>*	⇔	*member*	∈
star	⋆	*/<*	≮	*nomem*	∉
bigstar	✳	*/>*	≯	*cup*	∪
=dot	≐	*ang*	∠	*cap*	∩
orsign	∨	*rang*	∟	*incl*	⊆
andsign	∧	*3dot*	∴	*subset*	⊂
=del	≙	*thf*	∴	*supset*	⊃
oppA	∀	*quarter*	¼	*!subset*	⊈
oppE	∃	*3quarter*	¾	*!supset*	⊉
angstrom	Å	*degree*	°		

FILES

/usr/pub/eqnchar

SEE ALSO

eqn(1), **nroff**(1), **troff**(1)

NAME

filesystem – file system organization

SYNOPSIS

/
/usr

DESCRIPTION

The SunOS file system tree is organized for easy administration. Distinct areas within the file system tree are provided for files that are private to one machine, files that can be shared by multiple machines of a common architecture, files that can be shared by all machines, and home directories. This organization allows the sharable files to be stored on one machine, while being accessed by many machines using a remote file access mechanism such as Sun's Network File System (NFS). Grouping together similar files makes the file system tree easier to upgrade and manage.

The file system tree consists of a root file system and a collection of mountable file systems. The **mount**(8) program attaches mountable file systems to the file system tree at mount points (directory entries) in the root file system, or other previously mounted file systems. Two file systems, / (the root) and **/usr**, must be mounted in order to have a fully functional system. The root file system is mounted automatically by the kernel at boot time; the **/usr** file system is mounted by the **/etc/rc.boot** script, which is run as part of the booting process.

The root file system contains files that are unique to each machine; it can not be shared among machines. The root file system contains the following directories:

/dev Character and block special files. Device files provide hooks into hardware devices or operating system facilities. The **MAKEDEV** command (see **makedev**(8)) builds device files in the **/dev** directory. Typically, device files are built to match the kernel and hardware configuration of the machine.

/etc Various configuration files and system administration databases that are machine specific. You can think of **/etc** as the "home directory" of a machine, defining its "identity." Executable programs are no longer kept in **/etc**.

/home Mount points for home directories. This directory may be arranged so that shared user files are placed under the directory **/home/***machine-name* on machines serving as file servers. Machines may then be locally configured with mount points under **/home** for all of the file servers of interest, with the name of the mount point being the name of the file server.

/mnt A generic mount point. This is an empty directory available for temporarily mounting file systems on.

/sbin Executable programs that are needed in the boot process before **/usr** is mounted. **/sbin** contains *only* those programs that are needed in order to mount the **/usr** file system: **hostname**(1), **ifconfig**(8C), **init**(8), **mount**(8), and **sh**(1). After **/usr** is mounted, the full complement of utilities are available.

/tmp Temporary files that are deleted at reboot time.

/var Files, such as log files, that are unique to a machine but that can grow to an arbitrary ("variable") size.

/var/adm System logging and accounting files.

/var/preserve
Backup files for **vi**(1) and **ex**(1).

/var/spool Subdirectories for files used in printer spooling, mail delivery, **cron**(8), **at**(1), etc.

/var/tmp Transitory files that are not deleted at reboot time.

Because it is desirable to keep the root file system small, larger file systems are often mounted on **/var** and **/tmp**.

The file system mounted on **/usr** contains architecture-dependent and architecture-independent shareable files. The subtree rooted at **/usr/share** contains architecture-independent shareable files; the rest of the **/usr** tree contains architecture-dependent files. By mounting a common remote file system, a group of machines with a common architecture may share a single **/usr** file system. A single **/usr/share** file system can be shared by machines of any architecture. A machine acting as a file server may export many different **/usr** file systems to support several different architectures and operating system releases. Clients usually mount **/usr** read-only to prevent their accidentally modifying any shared files. The **/usr** file system contains the following subdirectories:

/usr/5bin	System V executables.
/usr/5include	System V include files.
/usr/5lib	System V library files.
/usr/bin	Executable programs. The bulk of the system utilities are located here.
/usr/dict	Dictionary databases.
/usr/etc	Executable system administration programs.
/usr/games	Executable game programs and data.
/usr/include	Include files.
/usr/lib	Program libraries and various architecture-dependent databases.
/usr/pub	Various data files.
/usr/ucb	Executable programs descended from the Berkeley Software Distribution.
/usr/share	Subtree for architecture-independent shareable files.
/usr/share/man	Subdirectories for the on-line reference manual pages.
/usr/share/lib	Architecture-independent databases.

A machine with disks may export root file systems, swap files and **/usr** file systems to diskless or partially-disked machines, which mount these into the standard file system hierarchy. The standard directory tree for exporting these file systems is:

/export	The root of the exported file system tree.
/export/exec/*architecture-name*	The exported **/usr** file system supporting *architecture-name* for the current release.
/export/exec/*architecture-name.release-name*	The exported **/usr** file system supporting *architecture-name* for SunOS *release-name*.
/export/share	The exported common **/usr/share** directory tree.
/export/root/*hostname*	The exported root file system for *hostname*.
/export/swap/*hostname*	The exported swap file for *hostname*.
/export/var/*hostname*	The exported **/var** directory tree for *hostname*.
/export/dump/*hostname*	The exported dump file for *hostname*.
/export/crash/*hostname*	The exported crash dump directory for *hostname*.

Changes from Previous Releases

The file system layout described here is quite a bit different from the layout employed previous to release 4.0 of SunOS. For compatibility with earlier releases of SunOS, and other versions of the UNIX system, symbolic links are provided for various files and directories linking their previous names to their current locations. The symbolic links provided include:

/bin —> /usr/bin	All programs previously located in **/bin** are now in **/usr/bin**.
/lib —> /usr/lib	All files previously located in **/lib** are now in **/usr/lib**.
/usr/adm —> /var/adm	The entire **/usr/adm** directory has been moved to **/var/adm**.
/usr/spool —> /var/spool	The entire **/usr/spool** directory has been moved to **/var/spool**.
/usr/tmp —> /var/tmp	The **/usr/tmp** directory has been moved to **/var/tmp**.

/etc/termcap —> /usr/share/lib/termcap

/usr/5lib/terminfo —> /usr/share/lib/terminfo

/usr/lib/me —> /usr/share/lib/me

/usr/lib/ms —> /usr/share/lib/ms

/usr/lib/tmac —> /usr/share/lib/tmac

/usr/man —> /usr/share/man

The following program binaries have been moved from **/etc** to **/usr/etc** with symbolic links to them left in **/etc**: **arp, clri, cron, chown, chroot, config, dkinfo, dmesg, dump, fastboot, fasthalt, fsck, halt, ifconfig, link, mkfs, mknod, mount, ncheck, newfs, pstat, rdump, reboot, renice, restore, rmt, rrestore, shutdown, umount, update, unlink**, and **vipw**.

In addition, some files and directories have been moved with no symbolic link left behind in the old location:

Old Name	*New Name*
/etc/biod	**/usr/etc/biod**
/etc/fsirand	**/usr/etc/fsirand**
/etc/getty	**/usr/etc/getty**
/etc/in.rlogind	**/usr/etc/in.rlogind**
/etc/in.routed	**/usr/etc/in.routed**
/etc/in.rshd	**/usr/etc/in.rshd**
/etc/inetd	**/usr/etc/inetd**
/etc/init	**/usr/etc/init**
/etc/nfsd	**/usr/etc/nfsd**
/etc/portmap	**/usr/etc/portmap**
/etc/rpc.lockd	**/usr/etc/rpc.lockd**
/etc/rpc.statd	**/usr/etc/rpc.statd**
/etc/ypbind	**/usr/etc/ypbind**
/usr/lib/sendmail.cf	**/etc/sendmail.cf**
/usr/preserve	**/var/preserve**
/usr/lib/aliases	**/etc/aliases**
/stand	**/usr/stand**
/etc/yp	**/var/yp**

Note: with this new file system organization, the approach to repairing a broken file system changes. One must mount **/usr** before doing an **fsck**(8), for example. If the mount point for **/usr** has been destroyed, **/usr** can be mounted temporarily on **/mnt** or **/tmp**. If the root file system on a standalone system is so badly damaged that none of these mount points exist, or if **/sbin/mount** has been corrupted, the only way to repair it may be to re-install the root file system.

SEE ALSO

at(1), **ex**(1), **hostname**(1), **sh**(1), **vi**(1), **intro**(4), **nfs**(4P), **hier**(7), **fsck**(8), **ifconfig**(8C), **init**(8), **makedev**(8), **mount**(8), **rc**(8)

NAME

hier – file system hierarchy

DESCRIPTION

The following outline gives a quick tour through a typical SunOS file system hierarchy:

/ root directory of the file system

/dev/ devices (Section 4)

MAKEDEV
shell script to create special files

MAKEDEV.local
site specific part of **MAKEDEV**

console main system console, **console**(4S)

drum paging device, **drum**(4)

***mem** memory special files, **mem**(4S)

null null file or data sink, **null**(4)

pty[p-z]*
pseudo terminal controllers, **pty**(4)

tty[ab] CPU serial ports, **zs**(4S)

tty[0123][0-f]
MTI serial ports **mti**(4S)

tty[hijk][0-f]
ALM-2 serial ports **mcp**(4S)

tty[p-z]*
pseudo terminals, **pty**(4)

vme* VME bus special files, **mem**(4S)

win window system special files, **win**(4S)

xy* disks, **xy**(4S)

rxy* raw disk interfaces, **xy**(4S)

...

/etc/ system-specific maintenance and data files

dumpdates
dump history, **dump**(8)

exports table of file systems exportable with NFS, **exports**(5)

fstab file system configuration table, **fstab**(5)

group group file, **group**(5)

hosts host name to network address mapping file, **hosts**(5)

hosts.equiv
list of trusted systems, **hosts.equiv**(5)

motd message of the day, **login**(1)

mtab mounted file table, **mtab**(5)

networks
network name to network number mapping file, **networks**(5)

passwd password file, **passwd**(5)

phones private phone numbers for remote hosts, as described in **phones**(5)

printcap
table of printers and capabilities, **printcap**(5)

protocols
protocol name to protocol number mapping file, **protocols**(5)

rc shell program to bring the system up multiuser

rc.boot startup file run at boot time

rc.local site dependent portion of **rc**

remote names and description of remote hosts for **tip**(1C), **remote**(5)

services
network services definition file, **services**(5)

ttytab database of terminal information used by **getty**(8)
...
/export/
directory of exported files and file systems for clients, including swap files, root, and **/usr** file systems
/home/ directory of mount points for remote-mounted home directories and shared file systems
user home (initial working) directory for *user*
.profile set environment for **sh**(1), **environ**(5V)
.project
what you are doing (used by (**finger**(1))
.cshrc startup file for **csh**(1)
.exrc startup file for **ex**(1)
.plan what your short-term plans are (used by **finger**(1))
.rhosts host equivalence file for **rlogin**(1C)
.mailrc startup file for **mail**(1)
calendar
user's datebook for **calendar**(1)
...
/lost+found
directory for connecting detached files for **fsck**(8)
/mnt/ mount point for file systems mounted temporarily
/sbin/ executable programs needed to mount **/usr/**
hostname
ifconfig
init
mount
sh
/tmp/ temporary files, usually on a fast device, see also **/var/tmp/**
ctm* used by **cc**(1V)
e* used by **ed**(1)
...
/var/ directory of files that tend to grow or vary in size
adm/ administrative log files
lastlog record of recent logins, **utmp**(5V)
lpacct line printer accounting **lpr**(1)
messages
system messages
tracct phototypesetter accounting, **troff**(1)
utmp table of currently logged in users, **utmp**(5V)
vaacct, vpacct
varian and versatec accounting **vtroff**(1), **pac**(8)
wtmp login history, **utmp**(5V)
...
preserve/
editor temporaries preserved here after crashes/hangups
spool/ delayed execution files
cron/ used by **cron**(8)
lpd/ used by **lpr**(1)
lock present when line printer is active
cf* copy of file to be printed, if necessary
df* control file for print job
tf* transient control file, while *lpr* is working

```
            mail/     mailboxes for mail(1)
                      name      mail file for user name
                      name.lock
                                lock file while name is receiving mail
            mqueue/
                      mail queue for sendmail(8)
            secretmail/
                      like mail/, but used by xsend(1)
            uucp/     work files and staging area for uucp(1C)
                      LOGFILE
                                summary log
                      LOG.*     log file for one transaction
            ...
      tmp/  temporary files, to keep /tmp/ small
            raster    used by plot(1G)
            stm*      used by sort(1V)
            ...
      yp/   Network Information Service (NIS) database files, ypfiles(5)
/usr/ general-purpose directory, usually a mounted file system
      bin/  utility programs
            as        assembler, as(1)
            cc        C compiler executive, c.f. /usr/lib/ccom, /usr/lib/cpp, /usr/lib/c2
            csh       the C-shell, csh(1)
            sh        the Bourne shell, sh(1)
            ...
      demo/ demonstration programs
      diag/ system tests and diagnostics
      dict/ word lists, etc.
            spellhist
                      history file for spell(1)
            words     principal word list, used by look(1)
            ...
      etc/  system administration programs; c.f. section 8
            catman    update preformatted man pages, catman(8)
            cron      the clock daemon, cron(8)
            dump      file system backup program dump(8)
            getty     part of login(1), getty(8)
            in.comsat
                      biff server (incoming mail daemon), comsat(8C)
            init      the parent of all processes, init(8)
            mount     mount(8)
            yp/       NIS programs
                      ypinit    build and install NIS database, ypinit(8)
                      yppush    force propagation of a changed NIS map, yppush(8)
                      ypset     point ypbind at a particular server, ypset(8)
                      ...
            ...
      games/
            backgammon
```

```
        lib/    library directory for game scores, etc.
                quiz.k/  what quiz(6) knows
                        africa   countries and capitals
                        index    category index
                        ...
                ...
        ...
hosts/  symbolic links to rsh(1C) for commonly accessed remote hosts
include/
        standard #include files
        a.out.h  object file layout, a.out(5)
        images/  icon images
        machine/
                header files from /usr/share/sys/sys/machine; may be a symbolic link
        math.h   intro(3M)
        net/     header files from /usr/share/sys/sys/net; may be a symbolic link
        nfs/     header files used in the Network File System (NFS)
        stdio.h  standard I/O, intro(3)
        sys/     kernel header files, c.f. /usr/share/sys/sys
        ...
lib/    object libraries, compiler program binaries, and other data
        ccom     C compiler proper
        cpp      C preprocessor
        c2       C code improver
        eign     list of English words to be ignored by ptx(1)
        font/    fonts for troff(1)
                ftR      Times Roman
                ftB      Times Bold
                ...
        libc.a   system calls, standard I/O, etc. (2,3,3S)
        libm.a   math library, intro(3M)
        lint/    utility files for lint
                lint[12]  subprocesses for lint(1V)
                llib-lc   dummy declarations for /usr/lib/libc.a, used by lint(1V)
                llib-lm   dummy declarations for /usr/lib/libm.a
                ...
        units    conversion tables for units(1)
        uucp/    programs and data for uucp(1C)
                L.sys    remote system names and numbers
                uucico   the real copy program
                ...
        ...
local/  locally maintained software
old/    obsolete and unsupported programs
pub/    publicly readable data files
sccs/   binaries of programs that compose the source code control system (SCCS)
src/    system source code tree
stand/  standalone programs (not run under the Sun Operating System)
share/  architecture independent files
        lib/     architecture independent data files
                termcap
                        description of terminal capabilities, termcap(5)
```

tmac/ macros for **troff**(1)
tmac.an
macros for **man**(7)
tmac.s macros for **ms**(7)
...
...
man/ on-line reference manual pages, **man**(1)
man?/ source files (**nroff**(1)) for sections 1 through 8 of the manual
as.1
...
cat?/ preformatted pages for sections 1 through 8 of the manual
...
sys/ SunOS kernel source and object modules
ucb/ binaries of programs developed at the University of California, Berkeley
ex line-oriented editor for experienced users, **ex**(1)
vi screen-oriented editor, **vi**(1)
...

/vmunix
the SunOS kernel binary

SEE ALSO

filesystem(7), **find**(1), **finger**(1), **grep**(1V), **ls**(1V), **rlogin**(1C), **whatis**(1), **whereis**(1), **which**(1), **ncheck**(8)

BUGS

The locations of files are subject to change without notice; the organization of your file system may vary.

This list is incomplete.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

iso_8859_1 – map of character set

SYNOPSIS

cat /usr/share/lib/locale/LC_CTYPE/iso_8859_1

DESCRIPTION

/usr/share/lib/locale/LC_CTYPE/iso_8859_1 is a map of the ISO_8859/1 character set, to be printed as needed.

This character set is available if **setlocale (3V)** is declared as:

setlocale(LC_CTYPE, iso_8859_1)

or:

setlocale(LC_ALL, iso_8859_1) see **setlocale**(3V) for more information about declaring categories and locales.

ISO Latin 1 Character Set

The following table displays the ISO 8859/1 character set.

ISO Latin 1				
Row/Col	Decimal	Octal		Name
02/00	032	040	SP	SPACE
02/01	033	041	!	EXCLAMATION POINT
02/02	034	042	"	QUOTATION MARK
02/03	035	043	#	NUMBER SIGN
02/04	036	044	$	DOLLAR SIGN
02/05	037	045	%	PERCENT SIGN
02/06	038	046	&	AMPERSAND
02/07	039	047	'	APOSTROPHE
02/08	040	050	(	LEFT PARENTHESIS
02/09	041	051	)	RIGHT PARENTHESIS
02/10	042	052	*	ASTERISK
02/11	043	053	+	PLUS SIGN
02/12	044	054	,	COMMA
02/13	045	055	-	HYPHEN, MINUS SIGN
02/14	046	056	.	FULL STOP (U.S.: PERIOD, DECIMAL POINT)
02/15	047	057	/	SOLIDUS (U.S.: SLASH)
03/00	048	060	0	DIGIT ZERO
03/01	049	061	1	DIGIT ONE
03/02	050	062	2	DIGIT TWO
03/03	051	063	3	DIGIT THREE
03/04	052	064	4	DIGIT FOUR
03/05	053	065	5	DIGIT FIVE
03/06	054	066	6	DIGIT SIX
03/07	055	067	7	DIGIT SEVEN
03/08	056	070	8	DIGIT EIGHT
03/09	057	071	9	DIGIT NINE
03/10	058	072	:	COLON
03/11	059	073		
03/12	060	074	<	LESS-THAN SIGN
03/13	061	075	=	EQUALS SIGN
03/14	062	076	>	GREATER-THAN SIGN
03/15	063	077	?	QUESTION MARK

ISO Latin 1 (continued)				
Row/Col	Decimal	Octal		Name
04/00	064	100	@	COMMERCIAL AT
04/01	065	101	A	LATIN CAPITAL LETTER A
04/02	066	102	B	LATIN CAPITAL LETTER B
04/03	067	103	C	LATIN CAPITAL LETTER C
04/04	068	104	D	LATIN CAPITAL LETTER D
04/05	069	105	E	LATIN CAPITAL LETTER E
04/06	070	106	F	LATIN CAPITAL LETTER F
04/07	071	107	G	LATIN CAPITAL LETTER G
04/08	072	110	H	LATIN CAPITAL LETTER H
04/09	073	111	I	LATIN CAPITAL LETTER I
04/10	074	112	J	LATIN CAPITAL LETTER J
04/11	075	113	K	LATIN CAPITAL LETTER K
04/12	076	114	L	LATIN CAPITAL LETTER L
04/13	077	115	M	LATIN CAPITAL LETTER M
04/14	078	116	N	LATIN CAPITAL LETTER N
04/15	079	117	O	LATIN CAPITAL LETTER O
05/00	080	120	P	LATIN CAPITAL LETTER P
05/01	081	121	Q	LATIN CAPITAL LETTER Q
05/02	082	122	R	LATIN CAPITAL LETTER R
05/03	083	123	S	LATIN CAPITAL LETTER S
05/04	084	124	T	LATIN CAPITAL LETTER T
05/05	085	125	U	LATIN CAPITAL LETTER U
05/06	086	126	V	LATIN CAPITAL LETTER V
05/07	087	127	W	LATIN CAPITAL LETTER W
05/08	088	130	X	LATIN CAPITAL LETTER X
05/09	089	131	Y	LATIN CAPITAL LETTER Y
05/10	090	132	Z	LATIN CAPITAL LETTER Z
05/11	091	133	[	LEFT SQUARE BRACKET
05/12	092	134	\	REVERSE SOLIDUS (U.S.: BACK SLASH)
05/13	093	135	]	RIGHT SQUARE BRACKET
05/14	094	136	^	CIRCUMFLEX ACCENT
05/15	095	137	_	LOW LINE (U.S.: UNDERSCORE)
06/00	096	140	`	GRAVE ACCENT
06/01	097	141	a	LATIN SMALL LETTER a
06/02	098	142	b	LATIN SMALL LETTER b
06/03	099	143	c	LATIN SMALL LETTER c
06/04	100	144	d	LATIN SMALL LETTER d
06/05	101	145	e	LATIN SMALL LETTER e
06/06	102	146	f	LATIN SMALL LETTER f
06/07	103	147	g	LATIN SMALL LETTER g
06/08	104	150	h	LATIN SMALL LETTER h
06/09	105	151	i	LATIN SMALL LETTER i
06/10	106	152	j	LATIN SMALL LETTER j
06/11	107	153	k	LATIN SMALL LETTER k
06/12	108	154	l	LATIN SMALL LETTER l
06/13	109	155	m	LATIN SMALL LETTER m
06/14	110	156	n	LATIN SMALL LETTER n
06/15	111	157	o	LATIN SMALL LETTER o

ISO Latin 1 (continued)				
Row/Col	Decimal	Octal		Name
07/00	112	160	p	LATIN SMALL LETTER p
07/01	113	161	q	LATIN SMALL LETTER q
07/02	114	162	r	LATIN SMALL LETTER r
07/03	115	163	s	LATIN SMALL LETTER s
07/04	116	164	t	LATIN SMALL LETTER t
07/05	117	165	u	LATIN SMALL LETTER u
07/06	118	166	v	LATIN SMALL LETTER v
07/07	119	167	w	LATIN SMALL LETTER w
07/08	120	170	x	LATIN SMALL LETTER x
07/09	121	171	y	LATIN SMALL LETTER y
07/10	122	172	z	LATIN SMALL LETTER z
07/11	123	173	{	LEFT CURLY BRACKET
07/12	124	174	|	VERTICAL LINE
07/13	125	175	}	RIGHT CURLY BRACKET
07/14	126	176	~	TILDE
10/00	160	240		NO-BREAK SPACE
10/01	161	241		INVERTED EXCLAMATION MARK
10/02	162	242		CENT SIGN
10/03	163	243		POUND SIGN
10/04	164	244		CURRENCY SIGN
10/05	165	245		YEN SIGN
10/06	166	246		BROKEN BAR
10/07	167	247		PARAGRAPH SIGN, (U.S.: SECTION SIGN)
10/08	168	250		DIAERESIS
10/09	169	251		COPYRIGHT SIGN
10/10	170	252		FEMININE ORDINAL INDICATOR
10/11	171	253		LEFT ANGLE QUOTATION MARK
10/12	172	254		NOT SIGN
10/13	173	255		SHY SOFT HYPHEN
10/14	174	256		REGISTERED TRADEMARK SIGN
10/15	175	257		MACRON
11/00	176	260		RING ABOVE, DEGREE SIGN
11/01	177	261		PLUS-MINUS SIGN
11/02	178	262		SUPERSCRIPT TWO
11/03	179	263		SUPERSCRIPT THREE
11/04	180	264		ACUTE ACCENT
11/05	181	265		MICRO SIGN
11/06	182	266		PILCROW SIGN, (U.S.: PARAGRAPH)
11/07	183	267		MIDDLE DOT
11/08	184	270		CEDILLA
11/09	185	271		SUPERSCRIPT ONE
11/10	186	272		MASCULINE ORDINAL INDICATOR
11/11	187	273		RIGHT ANGLE QUOTATION MARK
11/12	188	274		VULGAR FRACTION ONE QUARTER
11/13	189	275		VULGAR FRACTION ONE HALF
11/14	190	276		VULGAR FRACTION THREE QUARTERS
11/15	191	277		INVERTED QUESTION MARK

ISO Latin 1 (continued)			
Row/Col	Decimal	Octal	Name
12/00	192	300	LATIN CAPITAL LETTER A WITH GRAVE ACCENT
12/01	193	301	LATIN CAPITAL LETTER A WITH ACUTE ACCENT
12/02	194	302	LATIN CAPITAL LETTER A WITH CIRCUMFLEX ACCENT
12/03	195	303	LATIN CAPITAL LETTER A WITH TILDE
12/04	196	304	LATIN CAPITAL LETTER A WITH DIAERESIS
12/05	197	305	LATIN CAPITAL LETTER A WITH RING ABOVE
12/06	198	306	CAPITAL DIPHTHONG AE
12/07	199	307	LATIN CAPITAL LETTER C WITH CEDILLA
12/08	200	310	LATIN CAPITAL LETTER E WITH GRAVE ACCENT
12/09	201	311	LATIN CAPITAL LETTER E WITH ACUTE ACCENT
12/10	202	312	LATIN CAPITAL LETTER E WITH CIRCUMFLEX ACCENT
12/11	203	313	LATIN CAPITAL LETTER E WITH DIAERESIS
12/12	204	314	LATIN CAPITAL LETTER I WITH GRAVE ACCENT
12/13	205	315	LATIN CAPITAL LETTER I WITH ACUTE ACCENT
12/14	206	316	LATIN CAPITAL LETTER I WITH CIRCUMFLEX ACCENT
12/15	207	317	LATIN CAPITAL LETTER I WITH DIAERESIS
13/00	208	320	CAPITAL ICELANDIC LETTER ETH
13/01	209	321	LATIN CAPITAL LETTER N WITH TILDE
13/02	210	322	LATIN CAPITAL LETTER O WITH GRAVE ACCENT
13/03	211	323	LATIN CAPITAL LETTER O WITH ACUTE ACCENT
13/04	212	324	LATIN CAPITAL LETTER O WITH CIRCUMFLEX ACCENT
13/05	213	325	LATIN CAPITAL LETTER O WITH TILDE
13/06	214	326	LATIN CAPITAL LETTER O WITH DIAERESIS
13/07	215	327	MULTIPLICATION SIGN
13/08	216	330	LATIN CAPITAL LETTER O WITH OBLIQUE STROKE
13/09	217	331	LATIN CAPITAL LETTER U WITH GRAVE ACCENT
13/10	218	332	LATIN CAPITAL LETTER U WITH ACUTE ACCENT
13/11	219	333	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
13/12	220	334	LATIN CAPITAL LETTER U WITH DIAERESIS
13/13	221	335	LATIN CAPITAL LETTER Y WITH ACUTE ACCENT
13/14	222	336	CAPITAL ICELANDIC LETTER THORN
13/15	223	337	SMALL GERMAN LETTER SHARP s
14/00	224	340	LATIN SMALL LETTER a WITH GRAVE ACCENT
14/01	225	341	LATIN SMALL LETTER a WITH ACUTE ACCENT
14/02	226	342	LATIN SMALL LETTER a WITH CIRCUMFLEX ACCENT
14/03	227	343	LATIN SMALL LETTER a WITH TILDE
14/04	228	344	LATIN SMALL LETTER a WITH DIAERESIS
14/05	229	345	LATIN SMALL LETTER a WITH RING ABOVE
14/06	230	346	SMALL DIPHTHONG ae
14/07	231	347	LATIN SMALL LETTER c WITH CEDILLA
14/08	232	350	LATIN SMALL LETTER e WITH GRAVE ACCENT
14/09	233	351	LATIN SMALL LETTER e WITH ACUTE ACCENT
14/10	234	352	LATIN SMALL LETTER e WITH CIRCUMFLEX ACCENT
14/11	235	353	LATIN SMALL LETTER e WITH DIAERESIS
14/12	236	354	LATIN SMALL LETTER i WITH GRAVE ACCENT
14/13	237	355	LATIN SMALL LETTER i WITH ACUTE ACCENT
14/14	238	356	LATIN SMALL LETTER i WITH CIRCUMFLEX ACCENT
14/15	239	357	LATIN SMALL LETTER i WITH DIAERESIS

ISO Latin 1 (continued)			
Row/Col	Decimal	Octal	Name
15/00	240	360	SMALL ICELANDIC LETTER ETH
15/01	241	361	LATIN SMALL LETTER n WITH TILDE
15/02	242	362	LATIN SMALL LETTER o WITH GRAVE ACCENT
15/03	243	363	LATIN SMALL LETTER o WITH ACUTE ACCENT
15/04	244	364	LATIN SMALL LETTER o WITH CIRCUMFLEX ACCENT
15/05	245	365	LATIN SMALL LETTER o WITH TILDE
15/06	246	366	LATIN SMALL LETTER o WITH DIAERESIS
15/07	247	367	DIVISION SIGN
15/08	248	370	LATIN SMALL LETTER o WITH OBLIQUE STROKE
15/09	249	371	LATIN SMALL LETTER u WITH GRAVE ACCENT
15/10	250	372	LATIN SMALL LETTER u WITH ACUTE ACCENT
15/11	251	373	LATIN SMALL LETTER u WITH CIRCUMFLEX ACCENT
15/12	252	374	LATIN SMALL LETTER u WITH DIAERESIS
15/13	253	375	LATIN SMALL LETTER y WITH ACUTE ACCENT
15/14	254	376	SMALL ICELANDIC LETTER THORN
15/15	255	377	LATIN SMALL LETTER y WITH DIAERESIS

SEE ALSO

setlocale(3V)

NAME

man – macros to format Reference Manual pages

SYNOPSIS

nroff –man *filename...*

troff –man *filename...*

DESCRIPTION

These macros are used to lay out the reference pages in this manual. Note: if *filename* contains format input for a preprocessor, the commands shown above must be piped through the appropriate preprocessor. This is handled automatically by **man**(1). See **Conventions**.

Any text argument *t* may be zero to six words. Quotes may be used to include SPACE characters in a "word". If *text* is empty, the special treatment is applied to the next input line with text to be printed. In this way **.I** may be used to italicize a whole line, or **.SB** may be used to make small bold letters.

A prevailing indent distance is remembered between successive indented paragraphs, and is reset to default value upon reaching a non-indented paragraph. Default units for indents *i* are ens.

Type font and size are reset to default values before each paragraph, and after processing font and size setting macros.

These strings are predefined by **–man**:

*R '®', '(Reg)' in **nroff.**

*S Change to default type size.

Requests

Request	*Cause Break*	*If no Argument*	*Explanation*
.B *t*	no	*t*=n.t.l.*	Text is in bold font.
.BI *t*	no	*t*=n.t.l.	Join words, alternating bold and italic.
.BR *t*	no	*t*=n.t.l.	Join words, alternating bold and roman.
.DT	no	.5i 1i...	Restore default tabs.
.HP *i*	yes	*i*=p.i.*	Begin paragraph with hanging indent. Set prevailing indent to *i*.
.I *t*	no	*t*=n.t.l.	Text is italic.
.IB *t*	no	*t*=n.t.l.	Join words, alternating italic and bold.
.IP *x i*	yes	*x*=""	Same as **.TP** with tag *x*.
.IR *t*	no	*t*=n.t.l.	Join words, alternating italic and roman.
.IX *t*	no	-	Index macro, for Sun internal use.
.LP	yes	-	Begin left-aligned paragraph. Set prevailing indent to .5i.
.PD *d*	no	*d*=.4v	Set vertical distance between paragraphs.
.PP	yes	-	Same as **.LP**.
.RE	yes	-	End of relative indent. Restores prevailing indent.
.RB *t*	no	*t*=n.t.l.	Join words, alternating roman and bold.
.RI *t*	no	*t*=n.t.l.	Join words, alternating roman and italic.
.RS *i*	yes	*i*=p.i.	Start relative indent, increase indent by *i*. Sets prevailing indent to .5i for nested indents.
.SB *t*	no	-	Reduce size of text by 1 point, make text boldface.
.SH *t*	yes	-	Section Heading.
.SM *t*	no	*t*=n.t.l.	Reduce size of text by 1 point.
.SS *t*	yes	*t*=n.t.l.	Section Subheading.

.TH *n s d f m*	yes	-	Begin reference page *n*, of section *s*; *d* is the date of the most recent change. If present, *f* is the left page footer; *m* is the main page (center) header. Sets prevailing indent and tabs to .5i.
.TP *i*	yes	*i*=p.i.	Begin indented paragraph, with the tag given on the next text line. Set prevailing indent to *i*.
.TX *t p*	no	-	Resolve the title abbreviation *t*; join to punctuation mark (or text) *p*. *

n.t.l. = next text line; p.i. = prevailing indent

Conventions

When formatting a manual page, **man** examines the first line to determine whether it requires special processing. For example a first line consisting of:

```
'\" t
```

indicates that the manual page must be run through the **tbl**(1) preprocessor.

A typical manual page for a SunOS command or function is laid out as follows:

.TH *TITLE* [1-8]
The name of the command or function in upper-case, which serves as the title of the manual page. This is followed by the number of the section in which it appears.

.SH NAME The name, or list of names, by which the command is called, followed by a dash and then a one-line summary of the action performed. All in roman font, this section contains no **troff**(1) commands or escapes, and no macro requests. It is used to generate the **whatis**(1) database.

.SH SYNOPSIS

Commands:

The syntax of the command and its arguments, as typed on the command line. When in boldface, a word must be typed exactly as printed. When in italics, a word can be replaced with an argument that you supply. References to bold or italicized items are not capitalized in other sections, even when they begin a sentence.

Syntactic symbols appear in roman face:

[ ] An argument, when surrounded by brackets is optional.

| Arguments separated by a vertical bar are exclusive. You can supply only one item from such a list.

... Arguments followed by an ellipsis can be repeated. When an ellipsis follows a bracketed set, the expression within the brackets can be repeated.

Functions:

If required, the data declaration, or **#include** directive, is shown first, followed by the function declaration. Otherwise, the function declaration is shown.

.SH DESCRIPTION

A narrative overview of the command or function's external behavior. This includes how it interacts with files or data, and how it handles the standard input, standard output and standard error. Internals and implementation details are normally omitted. This section attempts to provide a succinct overview in answer to the question, "what does it do?"

Literal text from the synopsis appears in boldface, as do literal filenames and references to items that appear elsewhere in the *SunOS Reference Manual*. Arguments are italicized.

If a command interprets either subcommands or an input grammar, its command interface or input grammar is normally described in a **USAGE** section, which follows the **OPTIONS** section. The **DESCRIPTION** section only describes the behavior of the command itself, not that of subcommands.

.SH OPTIONS
The list of options along with a description of how each affects the command's operation.

.SH FILES
A list of files associated with the command or function.

.SH SEE ALSO
A comma-separated list of related manual pages, followed by references to other published materials.

.SH DIAGNOSTICS
A list of diagnostic messages and an explanation of each.

.SH BUGS
A description of limitations, known defects, and possible problems associated with the command or function.

FILES

/usr/share/lib/tmac/tmac.an

SEE ALSO

man(1), **nroff**(1), **troff**(1), **whatis**(1)

Formatting Documents.

NAME

me – macros for formatting papers

SYNOPSIS

nroff –me [options] file ...
troff –me [options] file ...

DESCRIPTION

This package of **nroff** and **troff** macro definitions provides a canned formatting facility for technical papers in various formats. When producing 2-column output on a terminal, filter the output through *col*(1).

The macro requests are defined below. Many **nroff** and **troff** requests are unsafe in conjunction with this package, however, these requests may be used with impunity after the first .pp:

.bp	begin new page
.br	break output line here
.sp n	insert n spacing lines
.ls n	(line spacing) n=1 single, n=2 double space
.na	no alignment of right margin
.ce n	center next n lines
.ul n	underline next n lines
.sz +n	add n to point size

Output of the **eqn, neqn, refer,** and **tbl**(1) preprocessors for equations and tables is acceptable as input.

REQUESTS

In the following list, "initialization" refers to the first .pp, .lp, .ip, .np, .sh, or .uh macro. This list is incomplete.

Request	Initial Value	Cause Break	Explanation
.(c	-	yes	Begin centered block
.(d	-	no	Begin delayed text
.(f	-	no	Begin footnote
.(l	-	yes	Begin list
.(q	-	yes	Begin major quote
.(x*x*	-	no	Begin indexed item in index *x*
.(z	-	no	Begin floating keep
.)c	-	yes	End centered block
.)d	-	yes	End delayed text
.)f	-	yes	End footnote
.)l	-	yes	End list
.)q	-	yes	End major quote
.)x	-	yes	End index item
.)z	-	yes	End floating keep
.++ *m H*	-	no	Define paper section. *m defines the part of the paper, and can be* **C** (chapter), **A** (appendix), **P** (preliminary, for instance, abstract, table of contents, etc.), **B** (bibliography), **RC** (chapters renumbered from page one each chapter), or **RA** (appendix renumbered from page one).
.+c *T*	-	yes	Begin chapter (or appendix, etc., as set by .++). *T* is the chapter title.
.1c	1	yes	One column format on a new page.
.2c	1	yes	Two column format.
.EN	-	yes	Space after equation produced by **eqn** or **meqn**.
.EQ *x y*	-	yes	Precede equation; break out and add space. Equation number is *y*. The optional argument *x* may be *I* to indent equation (default), *L* to left-adjust the equation, or *C* to center the equation.
.GE	-	yes	End *gremlin* picture.
.GS	-	yes	Begin *gremlin* picture.

.PE	-	yes	End *pic* picture.
.PS	-	yes	Begin *pic* picture.
.TE	-	yes	End table.
.TH	-	yes	End heading section of table.
.TS *x*	-	yes	Begin table; if *x* is *H* table has repeated heading.
.ac *A N*	-	no	Set up for ACM style output. *A* is the Author's name(s), *N* is the total number of pages. Must be given before the first initialization.
.b *x*	no	no	Print *x* in boldface; if no argument switch to boldface.
.ba +*n*	0	yes	Augments the base indent by *n*. This indent is used to set the indent on regular text (like paragraphs).
.bc	no	yes	Begin new column
.bi *x*	no	no	Print *x* in bold italics (nofill only)
.bu	-	yes	Begin bulleted paragraph
.bx *x*	no	no	Print *x* in a box (nofill only).
.ef 'x'y'z	''''	no	Set even footer to x y z
.eh 'x'y'z	''''	no	Set even header to x y z
.fo 'x'y'z	''''	no	Set footer to x y z
.hx	-	no	Suppress headers and footers on next page.
.he 'x'y'z	''''	no	Set header to x y z
.hl	-	yes	Draw a horizontal line
.i *x*	no	no	Italicize *x;* if *x* missing, italic text follows.
.ip *x y*	no	yes	Start indented paragraph, with hanging tag *x*. Indentation is *y* ens (default 5).
.lp	yes	yes	Start left-blocked paragraph.
.lo	-	no	Read in a file of local macros of the form .*x*. Must be given before initialization.
.np	1	yes	Start numbered paragraph.
.of 'x'y'z	''''	no	Set odd footer to x y z
.oh 'x'y'z	''''	no	Set odd header to x y z
.pd	-	yes	Print delayed text.
.pp	no	yes	Begin paragraph. First line indented.
.r	yes	no	Roman text follows.
.re	-	no	Reset tabs to default values.
.sc	no	no	Read in a file of special characters and diacritical marks. Must be given before initialization.
.sh *n x*	-	yes	Section head follows, font automatically bold. *n* is level of section, *x* is title of section.
.sk	no	no	Leave the next page blank. Only one page is remembered ahead.
.sm *x* -	*no*		*Set x* in a smaller pointsize.
.sz +*n*	10p	no	Augment the point size by *n* points.
.th	no	no	Produce the paper in thesis format. Must be given before initialization.
.tp	no	yes	Begin title page.
.u *x*	-	no	Underline argument (even in **troff**). (Nofill only).
.uh	-	yes	Like .sh but unnumbered.
.xp *x*	-	no	Print index *x*.

FILES

/usr/share/lib/tmac/tmac.e
/usr/share/lib/me/*

SEE ALSO

eqn(1), **nroff**(1), **troff**(1), **refer**(1), **tbl**(1)

Formatting Documents

NAME

ms – text formatting macros

SYNOPSIS

nroff –ms [*options*] *filename* ...

troff –ms [*options*] *filename* ...

DESCRIPTION

This package of **nroff**(1) and **troff**(1) macro definitions provides a formatting facility for various styles of articles, theses, and books. When producing 2-column output on a terminal or lineprinter, or when reverse line motions are needed, filter the output through **col**(1V). All external **–ms** macros are defined below.

Note: this **–ms** macro package is an extended version written at Berkeley and is a superset of the standard **–ms** macro packages as supplied by Bell Labs. Some of the Bell Labs macros have been removed; for instance, it is assumed that the user has little interest in producing headers stating that the memo was generated at Whippany Labs.

Many **nroff** and **troff** requests are unsafe in conjunction with this package. However, the first four requests below may be used with impunity after initialization, and the last two may be used even before initialization:

.bp	begin new page
.br	break output line
.sp *n*	insert n spacing lines
.ce *n*	center next n lines
.ls *n*	line spacing: *n*=1 single, *n*=2 double space
.na	no alignment of right margin

Font and point size changes with **\f** and **\s** are also allowed; for example, **\fIword\fR** will italicize *word*. Output of the **tbl**(1), **eqn**(1) and **refer**(1) preprocessors for equations, tables, and references is acceptable as input.

REQUESTS

Macro Name	Initial Value	Break? Reset?	Explanation
.AB *x*	–	y	begin abstract; if *x*=no do not label abstract
.AE	–	y	end abstract
.AI	–	y	author's institution
.AM	–	n	better accent mark definitions
.AU	–	y	author's name
.B *x*	–	n	embolden *x*; if no *x*, switch to boldface
.B1	–	y	begin text to be enclosed in a box
.B2	–	y	end boxed text and print it
.BT	date	n	bottom title, printed at foot of page
.BX *x*	–	n	print word *x* in a box
.CM	if t	n	cut mark between pages
.CT	–	y,y	chapter title: page number moved to CF (TM only)
.DA *x*	if n	n	force date *x* at bottom of page; today if no *x*
.DE	–	y	end display (unfilled text) of any kind
.DS *x y*	I	y	begin display with keep; *x*=I, L, C, B; *y*=indent
.ID *y*	8n,.5i	y	indented display with no keep; *y*=indent
.LD	–	y	left display with no keep
.CD	–	y	centered display with no keep
.BD	–	y	block display; center entire block
.EF *x*	–	n	even page footer *x* (3 part as for **.tl**)
.EH *x*	–	n	even page header *x* (3 part as for **.tl**)
.EN	–	y	end displayed equation produced by **eqn**

.EQ *x y*	–	y	break out equation; *x*=L,I,C; *y*=equation number
.FE	–	n	end footnote to be placed at bottom of page
.FP	–	n	numbered footnote paragraph; may be redefined
.FS *x*	–	n	start footnote; *x* is optional footnote label
.HD	undef	n	optional page header below header margin
.I *x*	–	n	italicize *x*; if no *x*, switch to italics
.IP *x y*	–	y,y	indented paragraph, with hanging tag *x*; *y*=indent
.IX *x y*	–	y	index words *x y* and so on (up to 5 levels)
.KE	–	n	end keep of any kind
.KF	–	n	begin floating keep; text fills remainder of page
.KS	–	y	begin keep; unit kept together on a single page
.LG	–	n	larger; increase point size by 2
.LP	–	y,y	left (block) paragraph.
.MC *x*	–	y,y	multiple columns; *x*=column width
.ND *x*	if t	n	no date in page footer; *x* is date on cover
.NH *x y*	–	y,y	numbered header; *x*=level, *x*=0 resets, *x*=S sets to *y*
.NL	10p	n	set point size back to normal
.OF *x*	–	n	odd page footer *x* (3 part as for **.tl**)
.OH *x*	–	n	odd page header *x* (3 part as for **.tl**)
.P1	if TM	n	print header on first page
.PP	–	y,y	paragraph with first line indented
.PT	- -	n	page title, printed at head of page
.PX *x*	–	y	print index (table of contents); *x*=no suppresses title
.QP	–	y,y	quote paragraph (indented and shorter)
.R	on	n	return to Roman font
.RE	5n	y,y	retreat: end level of relative indentation
.RP *x*	–	n	released paper format; *x*=no stops title on first page
.RS	5n	y,y	right shift: start level of relative indentation
.SH	–	y,y	section header, in boldface
.SM	–	n	smaller; decrease point size by 2
.TA	8n,5n	n	set TAB characters to 8n 16n ... (**nroff**) 5n 10n ... (**troff**)
.TC *x*	–	y	print table of contents at end; *x*=no suppresses title
.TE	–	y	end of table processed by **tbl**
.TH	–	y	end multi-page header of table
.TL	–	y	title in boldface and two points larger
.TM	off	n	UC Berkeley thesis mode
.TS *x*	–	y,y	begin table; if *x*=H table has multi-page header
.UL *x*	–	n	underline *x*, even in **troff**
.UX *x*	–	n	UNIX; trademark message first time; *x* appended
.XA *x y*	–	y	another index entry; *x*=page or no for none; *y*=indent
.XE	–	y	end index entry (or series of **.IX** entries)
.XP	–	y,y	paragraph with first line exdented, others indented
.XS *x y*	–	y	begin index entry; *x*=page or no for none; *y*=indent
.1C	on	y,y	one column format, on a new page
.2C	–	y,y	begin two column format
.]–	–	n	beginning of **refer** reference
.[0	–	n	end of unclassifiable type of reference
.[N	–	n	N= 1:journal-article, 2:book, 3:book-article, 4:report

REGISTERS

Formatting distances can be controlled in **–ms** by means of built-in number registers. For example, this sets the line length to 6.5 inches:

.nr LL 6.5i

Here is a table of number registers and their default values:

Name	Register Controls	Takes Effect	Default
PS	point size	paragraph	10
VS	vertical spacing	paragraph	12
LL	line length	paragraph	6i
LT	title length	next page	same as **LL**
FL	footnote length	next **.FS**	5.5i
PD	paragraph distance	paragraph	1v (if n), .3v (if t)
DD	display distance	displays	1v (if n), .5v (if t)
PI	paragraph indent	paragraph	5n
QI	quote indent	next **.QP**	5n
FI	footnote indent	next **.FS**	2n
PO	page offset	next page	0 (if n), ~1i (if t)
HM	header margin	next page	1i
FM	footer margin	next page	1i
FF	footnote format	next **.FS**	0 (1, 2, 3 available)

When resetting these values, make sure to specify the appropriate units. Setting the line length to 7, for example, will result in output with one character per line. Setting **FF** to 1 suppresses footnote superscripting; setting it to 2 also suppresses indentation of the first line; and setting it to 3 produces an **.IP**-like footnote paragraph.

Here is a list of string registers available in **–ms**; they may be used anywhere in the text:

Name	String's Function
***Q**	quote (" in **nroff**, `` in **troff**)
***U**	unquote (" in **nroff**, '' in **troff**)
***–**	dash (-- in **nroff**, — in **troff**)
***(MO**	month (month of the year)
***(DY**	day (current date)
******	automatically numbered footnote
***´**	acute accent (before letter)
***`**	grave accent (before letter)
***^**	circumflex (before letter)
***,**	cedilla (before letter)
***:**	umlaut (before letter)
***~**	tilde (before letter)

When using the extended accent mark definitions available with **.AM**, these strings should come after, rather than before, the letter to be accented.

FILES

/usr/share/lib/tmac/tmac.s
/usr/share/lib/ms/ms.???

SEE ALSO

col(1V), **eqn**(1), **nroff**(1), **refer**(1), **tbl**(1), **troff**(1)

Formatting Documents

BUGS

Floating keeps and regular keeps are diverted to the same space, so they cannot be mixed together with predictable results.

NAME

posix – overview of the IEEE Std 1003.1-1988 (POSIX.1) environment

SYNOPSIS

/usr/5bin/lint –n –lposix *posix_src.c*

AVAILABILITY

This environment is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

POSIX.1 is a set of functions and headers. The SunOS Release 4.1 implementation of POSIX.1 is a superset — it includes all of the functionality described in the IEEE standard as well as most of the SunOS functionality. See the **sunos(7)** man page for a description of SunOS functionality.

All man pages that are associated with POSIX.1 are marked by a "V" after the section number. Not all "V" pages, however, are POSIX.1. Some "V" pages may be part of other System V based environments such as X/Open.

If a user desires to work in a POSIX.1 (or System V) environment, the user should set the path variable to include **/usr/5bin** before anything else. The typical path is **PATH=/usr/5bin:/bin:/usr/bin:/usr/ucb**.

LINT

As a portability aid, Sun is providing a lint library that consists exclusively of POSIX.1 functions. Users may lint their code with the **–n –lposix** options to catch all non-POSIX.1 features.

POSIX.1 is primarily an operating system interface. POSIX.1 also specifies a subset of the functions defined by ANSI C. These are included in the **posix** lint library. Because of the additional functionality provided by ANSI C, Sun will also be providing an ANSI C (based on the December 7, 1988 draft) lint library. A portable application may want to lint with **–n –lposix –lansic** for the most complete coverage of functions.

POSIX.1 as with most other environments, continues to evolve. The **–lposix** lint library will always refer to the most recent standard supported by Sun. Some applications may wish to port to a particular version of the standard; they may safely use the more specific name of **–lposix1-88** (currently the same as **–lposix**).

Certain functions defined in the **posix** lint library are not available in the C library. In particular, math functions are made available only when the **–lm** option is added to **cc**(1V) or **ld**(1) commands.

FILES

/usr/5bin/*	POSIX.1 and System V specific executables
/usr/5include/*	POSIX.1 and System V specific headers
/usr/5lib/*	POSIX.1 and System V specific library files

SEE ALSO

lint(1V), **ansic**(7V), **bsd**(7), **sunos**(7), **svidii**(7V), **svidiii**(7V), **xopen**(7V)

IEEE Std 1003.1-1988

NAME

sunos, SunOS – overview of the SunOS Release 4.1 environment

SYNOPSIS

lint *sunos_src.c*

DESCRIPTION

The SunOS Release 4.1 lint library is a superset of the 4.3 BSD lint library. It includes all of the 4.3 BSD functionality, most of System V release 3.2 functionality, as well as extensive additional functionality in the networking and file system areas.

It is important to note that the default environment in SunOS Release 4.1 provides BSD 4.3 compatibility. Sun also provides a System V compatible environment (see **svidii**(7V)).

Note that many man pages are marked with a "V" after the section number, indicating some sort of System V compliance. SunOS functions are also documented on these man pages, as well as on man pages without the "V" section suffix.

By default, the user will get the SunOS environment. No path modifications should be necessary. The typical path is **set path = (/bin /usr/bin /usr/ucb)**

FILES

/usr/bin/*	SunOS executables
/usr/ucb/*	BSD derived executables
/usr/include/*	SunOS specific header files
/usr/lib/*	SunOS specific library files

SEE ALSO

lint(1V), **ansic**(7V), **bsd**(7), **posix**(7V), **svidii**(7V), **svidiii**(7V), **xopen**(7V)

NAME

svidii – overview of the System V environment

SYNOPSIS

/usr/5bin/lint –n –lsvidii *sys5_src.c*

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

SVID II is a set of functions and header files. The SunOS Release 4.1 implementation of SVID II is a superset — it includes all of the functionality described in the SVID issue 2 documents as well as most of the SunOS functionality. See the **sunos**(7) man page for a description of SunOS functionality.

All man pages that are associated with SVID II are marked by a "V" after the section number. Not all "V" pages are SVID II, however. Some "V" pages may be part of other System V based environments such as X/Open.

If a user desires to work in a SVID II environment, the user should set the path variable to include **/usr/xpg2bin** and **/usr/5bin** before anything else. The typical path is:

set path=(/usr/xpg2bin /usr/5bin /bin /usr/bin /usr/ucb)

As a portability aid, Sun is providing two lint libraries that consist exclusively of SVID II functions as defined in the SVID issue 2. Users may lint their code with the **–n –lsvidii** options to catch all features that are not found in SVID issue 2, all volumes. Using lint with the **–n –lsvidii-3** options is just like **–n –lsvidii** except that it does not include volume 3 (which contains new directory reading routines and new signal functions that appeared in System V release 3.2).

FILES

/usr/5bin/*	System V specific executables
/usr/5include/*	System V specific header files
/usr/5lib/*	System V specific library files

SEE ALSO

lint(1V), **ansic**(7V), **bsd**(7), **posix**(7V), **sunos**(7), **svidiii**(7V), **xopen**(7V)

NAME

svidiii – SVIDIII lint library

SYNOPSIS

/usr/5bin/lint –n –lsvidiii *svidiii_src.c*

AVAILABILITY

This environment is not fully tested under SunOS Release 4.1 as there is no test suite available. The environment that is believed to closely approximate a SVIDIII environment is the System V environment. The System V environment is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

SVIDIII is a future environment that Sun intends to support. SunOS Release 4.1 does not currently fully support SVIDIII applications. It does support many of the functions described by the SVIDIII document. This man page does not imply that the functions supported by SunOS Release 4.1 and the functions described by the SVIDIII document perform identically. The SVIDIII lint library is intended solely as a porting aid.

The SVIDIII lint library consists exclusively of SVIDIII functions. Users may lint their code with the **–n –lsvidiii** options to catch all non-SVIDIII features.

FILES

/usr/5lib/lint/llib-lsvidiii* SVIDIII C lint library

SEE ALSO

lint(1V), **ansic**(7V), **bsd**(7), **posix**(7V), **sunos**(7), **svidii**(7V), **xopen**(7V)

NAME

xopen – overview of the X/Open Portability Guide Issue 2 (X/Open) environment

SYNOPSIS

/usr/5bin/lint –n –lxopen *xopen_src.c*

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

X/Open is a set of functions and header files. The SunOS Release 4.1 implementation of X/Open is a superset — it includes all of the functionality described in the **/usr/group** Standard 1984 — as well as much of the System V functionality, and much of the SunOS functionality.

All man pages that are associated with X/Open are marked by a "V" after the section number. Not all "V" pages are X/Open, however. Some "V" pages may be part of other System V based environments such as POSIX.1.

If a user desires to work in a X/Open (or System V) environment, the user should set the path variable to include **/usr/xpg2bin** and **/usr/5bin** before anything else. The typical path is:

set path=(/usr/xpg2bin /usr/5bin /bin /usr/bin /usr/ucb)

As a portability aid, Sun is providing a lint library that consists exclusively of X/Open functions. Users may lint their code with the **–n –lxopen** options to catch all non-X/Open features.

X/Open, as with most other environments, continues to evolve. The **–lxopen** lint library will always refer to the most recent document supported by Sun. Some applications may wish to port to a particular version of the environment; they may safely use the more specific name of **–lxpg2** (currently the same as **–lxopen**).

FILES

/usr/xpg2bin/*	X/Open specific executables
/usr/xpg2include/*	X/Open specific header files
/usr/5include/*	System V specific header files
/usr/xpg2lib/*	X/Open specific library files
/usr/5lib/*	System V specific library files

SEE ALSO

lint(1V), **ansic**(7V), **bsd**(7), **posix**(7V), **sunos**(7), **svidii**(7V), **svidiii**(7V)

NAME

intro – introduction to system maintenance and operation commands

DESCRIPTION

This section contains information related to system bootstrapping, operation and maintenance. It describes all the server processes and daemons that run on the system, as well as standalone (PROM monitor) programs.

An 8V section number means one or more of the following:

- The man page documents System V behavior only.
- The man page documents default SunOS behavior, and System V behavior as it differs from the default behavior. These System V differences are presented under **SYSTEM V** section headers.
- The man page documents behavior compliant with *IEEE Std 1003.1-1988* (POSIX.1).

Disk formatting and labeling is done by **format**(8S). Bootstrapping of the system is described in **boot**(8S) and **init**(8). The standard set of commands run by the system when it boots is described in **rc**(8). Related commands include those that check the consistency of file systems, **fsck**(8); those that mount and unmount file systems, **mount**(8); add swap devices, **swapon**(8); force completion of outstanding file system I/O, **sync**(2); shutdown or reboot a running system **shutdown**(8), **halt**(8), and **reboot**(8); and, set the time on a machine from the time on another machine **rdate**(8C).

Creation of file systems is discussed in **mkfs**(8) and **newfs**(8). File system performance parameters can be adjusted with **tunefs**(8). File system backups and restores are described in **dump**(8) and **restore**(8).

Procedures for adding new users to a system are described in **adduser**(8), using **vipw**(8) to lock the password file during editing. **panic**(8S) which describes what happens when the system crashes, **savecore**(8) which can be used to analyze system crash dumps. Occasionally useful as adjuncts to the **fsck**(8) file system repair program are **clri**(8), **dcheck**(8), **icheck**(8), and **ncheck**(8).

Configuring a new version of the kernel requires using the program **config**(8); major system bootstraps often require the use of **mkproto**(8). New devices are added to the **/dev** directory (once device drivers are configured into the system) using **makedev**(8) and **mknod**(8). The **installboot**(8S) command can be used to install freshly compiled programs. The **catman**(8) command preformats the on-line manual pages.

Resource accounting is enabled by the **accton** command, and summarized by **sa**(8). Login time accounting is performed by **ac**(8). Disk quotas are managed using **quot**(8), **quotacheck**(8), **quotaon**(8), and **repquota**(8).

A number of servers and daemon processes are described in this section. The **update**(8) daemon forces delayed file system I/O to occur and **cron**(8) runs periodic events (such as removing temporary files from the disk periodically). The **syslogd**(8) daemon maintains the system error log. The **init**(8) process is the initial process created when the system boots. It manages the reboot process and creates the initial login prompts on the various system terminals, using **getty**(8). The Internet super-server **inetd**(8C) invokes all other internet servers as needed. These servers include the remote shell servers **rshd**(8C) and **rexecd**(8C), the remote login server **rlogind**(8C), the FTP and TELNET daemons **ftpd**(8C), and **telnetd**(8C), the TFTP daemon **tftpd**(8C), and the mail arrival notification daemon **comsat**(8C). Other network daemons include the 'load average/who is logged in' daemon **rwhod**(8C), the routing daemon **routed**(8C), and the mail daemon **sendmail**(8).

If network protocols are being debugged, then the protocol debugging trace program **trpt**(8C) is often useful. Remote magnetic tape access is provided by **rsh** and **rmt**(8C). Remote line printer access is provided by **lpd**(8), and control over the various print queues is provided by **lpc**(8). Printer cost-accounting is done through **pac**(8).

Network host tables may be gotten from the ARPA NIC using **gettable**(8C) and converted to UNIX-system-usable format using **htable**(8).

RPC and NFS daemons

RPC and NFS daemons include:

portmap	used by RPC based services.
ypbind	used by the Network Information Service (NIS) to locate the NIS server. Note: the Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.
biod	used by NFS clients to read ahead to, and write behind from, network file systems.
nfsd	the NFS server process that responds to NFS requests on NFS server machines.
ypserv	the NIS server, typically run on each NFS server.
rstatd	the server counterpart of the remote speedometer tools.
mountd	the **mount** server that runs on NFS server machines and responds to requests by other machines to mount file systems.
rwalld	used for broadcasting messages over the network.

LIST OF MAINTENANCE COMMANDS

Name	Appears on Page	Description
ac	**ac**(8)	login accounting
acctcms	**acctcms**(8)	command summary from per-process accounting records
acctcon1	**acctcon**(8)	connect-time accounting
acctcon2	**acctcon**(8)	connect-time accounting
acctdisk	**acct**(8)	miscellaneous accounting commands
acctdusg	**acct**(8)	miscellaneous accounting commands
acctmerg	**acctmerg**(8)	merge or add total accounting files
accton	**acct**(8)	miscellaneous accounting commands
accton	**sa**(8)	system accounting
acctprc1	**acctprc**(8)	process accounting
acctprc2	**acctprc**(8)	process accounting
acctwtmp	**acct**(8)	miscellaneous accounting commands
adbgen	**adbgen**(8)	generate adb script
add_client	**add_client**(8)	create a diskless network bootable NFS client on a server
add_services	**add_services**(8)	provide software installation services for any architecture
adduser	**adduser**(8)	procedure for adding new users
adv	**adv**(8)	advertise a directory for remote access with RFS
analyze	**old-analyze**(8)	postmortem system crash analyzer
arp	**arp**(8C)	address resolution display and control
audit	**audit**(8)	audit trail maintenance
auditd	**auditd**(8)	audit daemon
audit_warn	**audit_warn**(8)	audit daemon warning script
automount	**automount**(8)	automatically mount NFS file systems
biod	**nfsd**(8)	NFS daemons
boot	**boot**(8S)	start the system kernel, or a standalone program
bootparamd	**bootparamd**(8)	boot parameter server
C2conv	**c2conv**(8)	convert system to or from C2 security
C2unconv	**c2conv**(8)	convert system to or from C2 security
captoinfo	**captoinfo**(8V)	convert a termcap description into a terminfo description
catman	**catman**(8)	create the cat files for the manual
change_login	**change_login**(8)	control screen blanking and choice of login utility
chargefee	**acctsh**(8)	shell procedures for accounting
check4	**set4**(8)	check the virtual address space limit flag in a module
chown	**chown**(8)	change owner
chroot	**chroot**(8)	change root directory for a command
chrtbl	**chrtbl**(8)	generate character classification table
ckpacct	**acctsh**(8)	shell procedures for accounting

client	**client**(8)	add or remove diskless Sun386i systems
clri	**clri**(8)	clear inode
colldef	**colldef**(8)	convert collation sequence source definition
comsat	**comsat**(8C)	biff server
config	**config**(8)	build system configuration files
copy_home	**copy_home**(8)	fetch default startup files for new home directories
crash	**crash**(8)	examine system images
cron	**cron**(8)	clock daemon
dbconfig	**dbconfig**(8)	initializes the dial box
dcheck	**dcheck**(8)	file system directory consistency check
devinfo	**devinfo**(8S)	print out system device information
devnm	**devnm**(8V)	device name
diskusg	**diskusg**(8)	generate disk accounting data by user
dkctl	**dkctl**(8)	control special disk operations
dkinfo	**dkinfo**(8)	report information about a disk's geometry and partitioning
dmesg	**dmesg**(8)	collect system diagnostic messages to form error log
dname	**dname**(8)	print RFS domain and network names
dodisk	**acctsh**(8)	shell procedures for accounting
dorfs	**dorfs**(8)	initialize, start and stop RFS automatically
dump	**dump**(8)	incremental file system dump
dumpfs	**dumpfs**(8)	dump file system information
edquota	**edquota**(8)	edit user quotas
eeprom	**eeprom**(8S)	EEPROM display and load utility
etherd	**etherd**(8C)	Ethernet statistics server
etherfind	**etherfind**(8C)	find packets on Ethernet
exportfs	**exportfs**(8)	export and unexport directories to NFS clients
extract_unbundled	**extract_unbundled**(8)	extract and execute unbundled-product installation scripts
fastboot	**fastboot**(8)	reboot/halt the system without checking the disks
fasthalt	**fastboot**(8)	reboot/halt the system without checking the disks
fingerd	**fingerd**(8C)	remote user information server
format	**format**(8S)	disk partitioning and maintenance utility
fpa_download	**fpa_download**(8)	download to the Floating Point Accelerator
fparel	**fparel**(8)	Sun FPA online reliability tests
fpaversion	**fpaversion**(8)	print FPA version, load microcode
fpurel	**fpurel**(8)	perform tests the Sun Floating Point Co-processor
fpuversion4	**fpuversion4**(8)	print the Sun-4 FPU version
fsck	**fsck**(8)	file system consistency check and interactive repair
fsirand	**fsirand**(8)	install random inode generation numbers
ftpd	**ftpd**(8C)	TCP/IP Internet File Transfer Protocol server
fumount	**fumount**(8)	force unmount of an advertised RFS resource
fusage	**fusage**(8)	RFS disk access profiler
fuser	**fuser**(8)	identify processes using a file or file structure
fwtmp	**fwtmp**(8)	manipulate connect accounting records
gettable	**gettable**(8C)	get DARPA Internet format host table from a host
getty	**getty**(8)	set terminal mode
gid_allocd	**uid_allocd**(8C)	UID and GID allocator daemons
gpconfig	**gpconfig**(8)	initialize the Graphics Processor
grpck	**grpck**(8V)	check group database entries
gxtest	**gxtest**(8S)	stand alone test for the Sun video graphics board
halt	**halt**(8)	stop the processor
hostrfs	**hostrfs**(8)	Convert IP addresses to RFS format
htable	**htable**(8)	convert DoD Internet format host table
icheck	**icheck**(8)	file system storage consistency check

idload	**idload**(8)	RFS user and group mapping
ifconfig	**ifconfig**(8C)	configure network interface parameters
imemtest	**imemtest**(8S)	stand alone memory test
in.comsat	**comsat**(8C)	biff server
inetd	**inetd**(8C)	Internet services daemon
in.fingerd	**fingerd**(8C)	remote user information server
infocmp	**infocmp**(8V)	compare or print out terminfo descriptions
in.ftpd	**ftpd**(8C)	TCP/IP Internet File Transfer Protocol server
init	**init**(8)	process control initialization
in.named	**named**(8C)	Internet domain name server
in.rexecd	**rexecd**(8C)	remote execution server
in.rlogind	**rlogind**(8C)	remote login server
in.routed	**routed**(8C)	network routing daemon
in.rshd	**rshd**(8C)	remote shell server
in.rwhod	**rwhod**(8C)	system status server
installboot	**installboot**(8S)	install bootblocks in a disk partition
install_small_kernel	**install_small_kernel**(8)	install a small, pre-configured kernel
installtxt	**installtxt**(8)	create a message archive
in.talkd	**talkd**(8C)	server for talk program
in.telnetd	**telnetd**(8C)	TCP/IP TELNET protocol server
in.tftpd	**tftpd**(8C)	TCP/IP Trivial File Transfer Protocol server
in.tnamed	**tnamed**(8C)	TCP/IP Trivial name server
intr	**intr**(8)	allow a command to be interruptible
iostat	**iostat**(8)	report I/O statistics
ipallocd	**ipallocd**(8C)	Ethernet-to-IP address allocator
kadb	**kadb**(8S)	adb-like kernel and standalone-program debugger
keyenvoy	**keyenvoy**(8C)	talk to keyserver
keyserv	**keyserv**(8C)	server for storing public and private keys
kgmon	**kgmon**(8)	generate a dump of the operating system's profile buffers
lastlogin	**acctsh**(8)	shell procedures for accounting
ldconfig	**ldconfig**(8)	link-editor configuration
link	**link**(8V)	exercise link and unlink system calls
listen	**nlsadmin**(8)	network listener service administration for RFS
lockd	**lockd**(8C)	network lock daemon
logintool	**logintool**(8)	graphic login interface
lpc	**lpc**(8)	line printer control program
lpd	**lpd**(8)	printer daemon
mailstats	**mailstats**(8)	print statistics collected by sendmail
makedbm	**makedbm**(8)	make a NIS ndbm file
MAKEDEV	**makedev**(8)	make system special files
makekey	**makekey**(8)	generate encryption key
mc68881version	**mc68881version**(8)	print the MC68881 mask number and approximate clock rate
mconnect	**mconnect**(8)	connect to SMTP mail server socket
mkfile	**mkfile**(8)	create a file
mkfs	**mkfs**(8)	construct a file system
mknod	**mknod**(8)	build special file
mkproto	**mkproto**(8)	construct a prototype file system
modload	**modload**(8)	load a module
modstat	**modstat**(8)	display status of loadable modules
modunload	**modunload**(8)	unload a module
monacct	**acctsh**(8)	shell procedures for accounting
monitor	**monitor**(8S)	system ROM monitor
mountd	**mountd**(8C)	NFS mount request server

mount	**mount**(8)	mount and unmount file systems
mount_tfs	**mount_tfs**(8)	mount and dismount TFS filesystems
named	**named**(8C)	Internet domain name server
ncheck	**ncheck**(8)	generate names from i-numbers
ndbootd	**ndbootd**(8C)	ND boot block server
netconfig	**netconfig**(8C)	PNP boot service
netstat	**netstat**(8C)	show network status
newaliases	**newaliases**(8)	rebuild the data base for the mail aliases file
newfs	**newfs**(8)	create a new file system
newkey	**newkey**(8)	create a new key in the publickey database
nfsd	**nfsd**(8)	NFS daemons
nfsstat	**nfsstat**(8C)	Network File System statistics
nlsadmin	**nlsadmin**(8)	network listener service administration for RFS
nslookup	**nslookup**(8C)	query domain name servers interactively
nsquery	**nsquery**(8)	RFS name server query
nulladm	**acctsh**(8)	shell procedures for accounting
old-analyze	**old-analyze**(8)	postmortem system crash analyzer
pac	**pac**(8)	printer/plotter accounting information
panic	**panic**(8S)	what happens when the system crashes
ping	**ping**(8C)	send ICMP ECHO_REQUEST packets to network hosts
pnpboot	**pnpboot**(8C)	pnp diskless boot service
pnpd	**pnpd**(8C)	PNP daemon
pnp.s386	**pnpboot**(8C)	pnp diskless boot service
portmap	**portmap**(8C)	TCP/IP port to RPC program number mapper
praudit	**praudit**(8)	print contents of an audit trail file
prctmp	**acctsh**(8)	shell procedures for accounting
prdaily	**acctsh**(8)	shell procedures for accounting
prtacct	**acctsh**(8)	shell procedures for accounting
pstat	**pstat**(8)	print system facts
pwck	**pwck**(8V)	check password database entries
pwdauthd	**pwdauthd**(8C)	server for authenticating passwords
quotacheck	**quotacheck**(8)	file system quota consistency checker
quotaoff	**quotaon**(8)	turn file system quotas on and off
quotaon	**quotaon**(8)	turn file system quotas on and off
quot	**quot**(8)	summarize file system ownership
rarpd	**rarpd**(8C)	TCP/IP Reverse Address Resolution Protocol server
rc	**rc**(8)	command scripts for auto-reboot and daemons
rc.boot	**rc**(8)	command scripts for auto-reboot and daemons
rc.local	**rc**(8)	command scripts for auto-reboot and daemons
rdate	**rdate**(8C)	set system date from a remote host
rdump	**dump**(8)	incremental file system dump
reboot	**reboot**(8)	restart the operating system
renice	**renice**(8)	alter nice value of running processes
repquota	**repquota**(8)	summarize quotas for a file system
restore	**restore**(8)	incremental file system restore
rexd	**rexd**(8C)	RPC-based remote execution server
rexecd	**rexecd**(8C)	remote execution server
rfadmin	**rfadmin**(8)	RFS domain administration
rfpasswd	**rfpasswd**(8)	change RFS host password
rfstart	**rfstart**(8)	start RFS
rfstop	**rfstop**(8)	stop the RFS environment
rfuadmin	**rfuadmin**(8)	RFS notification shell script
rfudaemon	**rfudaemon**(8)	Remote File Sharing daemon

rlogind	**rlogind**(8C)	remote login server
rmail	**rmail**(8C)	handle remote mail received via uucp
rm_client	**rm_client**(8)	remove an NFS client
rmntstat	**rmntstat**(8)	display RFS mounted resource information
rmt	**rmt**(8C)	remote magtape protocol module
route	**route**(8C)	manually manipulate the routing tables
routed	**routed**(8C)	network routing daemon
rpc.etherd	**etherd**(8C)	Ethernet statistics server
rpcinfo	**rpcinfo**(8C)	report RPC information
rpc.lockd	**lockd**(8C)	network lock daemon
rpc.mountd	**mountd**(8C)	NFS mount request server
rpc.rexd	**rexd**(8C)	RPC-based remote execution server
rpc.rquotad	**rquotad**(8C)	remote quota server
rpc.rstatd	**rstatd**(8C)	kernel statistics server
rpc.rusersd	**rusersd**(8C)	network username server
rpc.rwalld	**rwalld**(8C)	network rwall server
rpc.sprayd	**sprayd**(8C)	spray server
rpc.statd	**statd**(8C)	network status monitor
rpc.yppasswdd	**yppasswdd**(8C)	server for modifying NIS password file
rpc.ypupdated	**ypupdated**(8C)	server for changing NIS information
rquotad	**rquotad**(8C)	remote quota server
rrestore	**restore**(8)	incremental file system restore
rshd	**rshd**(8C)	remote shell server
rstatd	**rstatd**(8C)	kernel statistics server
runacct	**acctsh**(8)	shell procedures for accounting
runacct	**runacct**(8)	run daily accounting
rusage	**rusage**(8)	print resource usage for a command
rusersd	**rusersd**(8C)	network username server
rwalld	**rwalld**(8C)	network rwall server
rwhod	**rwhod**(8C)	system status server
sa	**sa**(8)	system accounting
savecore	**savecore**(8)	save a core dump of the operating system
sendmail	**sendmail**(8)	send mail over the internet
set4	**set4**(8)	set the virtual address space limit flag in a module
setsid	**setsid**(8V)	set process to session leader
showfhd	**showfhd**(8C)	showfh daemon run on the NFS servers
showfh	**showfh**(8C)	print full pathname of file from the NFS file handle
showmount	**showmount**(8)	show all remote mounts
shutacct	**acctsh**(8)	shell procedures for accounting
shutdown	**shutdown**(8)	close down the system at a given time
skyversion	**skyversion**(8)	print the SKYFFP board microcode version number
sprayd	**sprayd**(8C)	spray server
spray	**spray**(8C)	spray packets
start_applic	**start_applic**(8)	generic application startup procedures
startup	**acctsh**(8)	shell procedures for accounting
statd	**statd**(8C)	network status monitor
sticky	**sticky**(8)	mark files for special treatment
sundiag	**sundiag**(8)	system diagnostics
suninstall	**suninstall**(8)	install and upgrade the SunOS operating system
swapon	**swapon**(8)	specify additional device for paging and swapping
sys-config	**sys-config**(8)	configure a system or administer configuration information
syslogd	**syslogd**(8)	log system messages
sys-unconfig	**sys-unconfig**(8)	undo a system's configuration

talkd	**talkd**(8C)	server for talk program
telnetd	**telnetd**(8C)	TCP/IP TELNET protocol server
tfsd	**tfsd**(8)	TFS daemon
tftpd	**tftpd**(8C)	TCP/IP Trivial File Transfer Protocol server
tic	**tic**(8V)	terminfo compiler
tnamed	**tnamed**(8C)	TCP/IP Trivial name server
trpt	**trpt**(8C)	transliterate protocol trace
ttysoftcar	**ttysoftcar**(8)	enable/disable carrier detect
tunefs	**tunefs**(8)	tune up an existing file system
turnacct	**acctsh**(8)	shell procedures for accounting
tzsetup	**tzsetup**(8)	set up old-style time zone information in the kernel
uid_allocd	**uid_allocd**(8C)	UID and GID allocator daemons
umount	**mount**(8)	mount and unmount file systems
umount_tfs	**mount_tfs**(8)	mount and dismount TFS filesystems
unadv	**unadv**(8)	unadvertise a Remote File Sharing resource
unconfigure	**unconfigure**(8)	reset the network configuration for a Sun386i system
unlink	**link**(8V)	exercise link and unlink system calls
unset4	**set4**(8)	unset the virtual address space limit flag in a module
update	**update**(8)	periodically update the super block
user_agentd	**user_agentd**(8C)	user agent daemon
uucheck	**uucheck**(8C)	check the UUCP directories and Permissions file
uucico	**uucico**(8C)	file transport program for the UUCP system
uuclean	**uuclean**(8C)	uucp spool directory clean-up
uucleanup	**uucleanup**(8C)	UUCP spool directory clean-up
uucpd	**uucpd**(8C)	UUCP server
uusched	**uusched**(8C)	the scheduler for the UUCP file transport program
uuxqt	**uuxqt**(8C)	execute remote command requests
vipw	**vipw**(8)	edit the password file
vmstat	**vmstat**(8)	report virtual memory statistics
wtmpfix	**fwtmp**(8)	manipulate connect accounting records
ypbatchupd	**ypbatchupd**(8C)	NIS batch update daemon
ypbind	**ypserv**(8)	NIS server and binder processes
ypinit	**ypinit**(8)	build and install NIS database
ypmake	**ypmake**(8)	rebuild NIS database
yppasswdd	**yppasswdd**(8C)	server for modifying NIS password file
yppoll	**yppoll**(8)	version of NIS map at NIS server
yppush	**yppush**(8)	force propagation of changed NIS map
ypserv	**ypserv**(8)	NIS server and binder processes
ypset	**ypset**(8)	point ypbind at a particular server
ypsync	**ypsync**(8)	collect most up-to-date NIS maps
ypupdated	**ypupdated**(8C)	server for changing NIS information
ypxfr	**ypxfr**(8)	transfer NIS map from NIS server to here
zdump	**zdump**(8)	time zone dumper
zic	**zic**(8)	time zone compiler

NAME

ac – login accounting

SYNOPSIS

/usr/etc/ac [**–w** *wtmp*] [**–p**] [**–d**] [*username*] ...

DESCRIPTION

ac produces a printout giving connect time for each user who has logged in during the life of the current *wtmp* file. A total is also produced.

The accounting file **/var/adm/wtmp** is maintained by **init**(8) and **login**(1). Neither of these programs creates the file, so if it does not exist no connect-time accounting is done. To start accounting, it should be created with length 0. On the other hand if the file is left undisturbed it will grow without bound, so periodically any information desired should be collected and the file truncated.

OPTIONS

–w *wtmp*
Specify an alternate *wtmp* file.

–p Print individual totals; without this option, only totals are printed.

–d Printout for each midnight to midnight period. Any *people* will limit the printout to only the specified login names. If no *wtmp* file is given, **/var/adm/wtmp** is used.

FILES

/var/adm/wtmp

SEE ALSO

login(1), **utmp**(5V), **init**(8), **sa**(8)

NAME

acctdisk, acctdusg, accton, acctwtmp – overview of accounting and miscellaneous accounting commands

SYNOPSIS

/usr/lib/acct/acctdisk

/usr/lib/acct/acctdusg [**–u** *filename*] [**–p** *filename*]

/usr/lib/acct/accton [*filename*]

/usr/lib/acct/acctwtmp *reason*

DESCRIPTION

Accounting software is structured as a set of tools (consisting of both C programs and shell procedures) that can be used to build accounting systems. **acctsh**(8) describes the set of shell procedures built on top of the C programs.

Connect time accounting is handled by various programs that write records into **/etc/utmp**, as described in **utmp**(5V). The programs described in **acctcon**(8) convert this file into session and charging records, which are then summarized by **acctmerg**(8).

Process accounting is performed by the UNIX system kernel. Upon termination of a process, one record per process is written to a file (normally **/var/adm/pacct**). The programs in **acctprc**(8) summarize this data for charging purposes; **acctcms**(8) is used to summarize command usage. Current process data may be examined using **acctcom**(1).

Process accounting and connect time accounting (or any accounting records in the format described in **acct**(5)) can be merged and summarized into total accounting records by **acctmerg** (see **tacct** format in **acct**(5)). **prtacct** (see **acctsh**(8)) is used to format any or all accounting records.

acctdisk reads lines that contain user ID, login name, and number of disk blocks and converts them to total accounting records that can be merged with other accounting records.

acctdusg reads its standard input (usually from '**find / –print**') and computes disk resource consumption (including indirect blocks) by login.

accton without arguments turns process accounting off. If *filename* is given, it must be the name of an existing file, to which the kernel appends process accounting records (see **acct**(2V) and **acct**(5)). You must be super-user to use this command.

acctwtmp writes a **utmp**(5V) record to its standard output. The record contains the current time and a string of characters that describe the *reason*. The login name for this record is set to **@@acct** (see **utmp**(5V)). *reason* must be a string of 8 or fewer characters, numbers, **$**, or SPACE characters. If *reason* contains a SPACE character, it must be enclosed in double quotes. For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

```
acctwtmp uname >> /var/adm/wtmp
acctwtmp fsave >> /var/adm/wtmp
```

OPTIONS

acctdusg

–u *filename*

Place records consisting of those file names for which **acctdusg** charges no one in *filename* (a potential source for finding users trying to avoid disk charges).

–p *filename*

Use *filename* as the password file, rather than **/etc/passwd**. (See **diskusg**(8) for more details.)

FILES

/etc/passwd	used for login name to user ID conversions
/usr/lib/acct	holds all accounting commands listed in section 8 of this manual
/var/adm/pacct	current process accounting file
/var/adm/wtmp	login/logoff history file

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **diskusg**(8), **fwtmp**(8), **runacct**(8)

NAME

acctcms – command summary from per-process accounting records

SYNOPSIS

/usr/lib/acct/acctcms [**–cjnst**] *filename* ...

/usr/lib/acct/acctcms [**–a** [**po**] [**cjnstpo**] *filename* ...

DESCRIPTION

acctcms reads one or more *filename*s, normally in the form described in **acct**(5). It adds all records for processes that executed identically-named commands, sorts them, and writes them to the standard output, normally using an internal summary format.

OPTIONS

–a Print output in ASCII rather than in the internal summary format. The output includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, "hog factor", characters transferred, and blocks read and written, as in **acctcom**(1). Output is normally sorted by total kcore-minutes.

–c Sort by total CPU time, rather than total kcore-minutes.

–j Combine all commands invoked only once under "***other".

–n Sort by number of processes.

–s Any file names encountered hereafter are already in internal summary format.

–t Process all records as total accounting records. The default internal summary format splits each field into prime and non-prime time parts. This option combines the prime and non-prime time parts into a single field that is the total of both.

The following options may be used only with the **–a** option.

–p Output a prime-time-only command summary.

–o Output a non-prime (offshift) time only command summary.

When **–p** and **–o** are used together, a combination prime and non-prime time report is produced. All the output summaries will be total usage except number of times executed, CPU minutes, and real minutes which will be split into prime and non-prime.

EXAMPLES

A typical sequence for performing daily command accounting and for maintaining a running total is:

```
acctcms file ... >today
cp total previoustotal
acctcms –s today previoustotal >total
acctcms –a –s today
```

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **fwtmp**(8), **runacct**(8)

BUGS

Unpredictable output results if **–t** is used on new style internal summary format files, or if it is not used with old style internal summary format files.

NAME

acctcon1, acctcon2 – connect-time accounting

SYNOPSIS

/usr/lib/acct/acctcon1 [**–pt**] [**–l** *file*] [**–o** *file*]

/usr/lib/acct/acctcon2

DESCRIPTION

acctcon1

acctcon1 converts a sequence of login/logoff records read from its standard input to a sequence of records, one per login session. Its input should normally be redirected from **/var/adm/wtmp**. Its output is ASCII, giving device, user ID, login name, prime connect time (seconds), non-prime connect time (seconds), session starting time (numeric), and starting date and time.

acctcon2

acctcon2 expects as input a sequence of login session records and converts them into total accounting records (see **tacct** format in **acct**(5)).

OPTIONS

acctcon1

–p Print input only, showing line name, login name, and time (in both numeric and date/time formats).

–t Test mode. **acctcon1** maintains a list of lines on which users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress. The **–t** flag causes it to use, instead, the last time found in its input, thus assuring reasonable and repeatable numbers for non-current files.

–l *file* *file* is created to contain a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware oddities. Hang-up, termination of **login**(1) and termination of the login shell each generate logoff records, so that the number of logoffs is often three to four times the number of sessions. See **init**(8) and **utmp**(5V).

–o *file* *file* is filled with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

EXAMPLES

These commands are typically used as shown below. The file **ctmp** is created only for the use of **acctprc**(8) commands:

```
acctcon1 –t –l lineuse –o reboots <wtmp | sort +1n +2 >ctmp
acctcon2 <ctmp | acctmerg >ctacct
```

FILES

/var/adm/wtmp

SEE ALSO

acctcom(1), **login**(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **fwtmp**(8), **init**(8), **runacct**(8)

BUGS

The line usage report is confused by date changes. Use **wtmpfix** (see **fwtmp**(8)) to correct this situation.

NAME

acctmerg – merge or add total accounting files

SYNOPSIS

/usr/lib/acct/acctmerg [**–aiptuv**] [*filename*...]

DESCRIPTION

acctmerg reads its standard input and up to nine additional files, all in the **tacct** format (see **acct**(5)) or an ASCII version thereof. It merges these inputs by adding records whose keys (normally user ID and name) are identical, and expects the inputs to be sorted on those keys.

OPTIONS

–a Produce output in ASCII version of **tacct**.

–i Input files are in ASCII version of **tacct**.

–p Print input with no processing.

–t Produce a single record that totals all input.

–u Summarize by user ID, rather than user ID and name.

–v Produce output in verbose ASCII format, with more precise notation for floating point numbers.

EXAMPLES

The following sequence is useful for making "repairs" to any file kept in this format:

```
acctmerg –v <filename1 >filename2
        edit file2 as desired ...
acctmerg –i <filename2 >filename1
```

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctprc**(8), **acctsh**(8), **fwtmp**(8), **runacct**(8)

NAME

acctprc1, acctprc2 – process accounting

SYNOPSIS

/usr/lib/acct/acctprc1 [*ctmp*]

/usr/lib/acct/acctprc2

DESCRIPTION

acctprc1

acctprc1 reads input in the form described by **acct**(5), adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (ticks), non-prime CPU time (ticks), and mean memory size (in pages). If *ctmp* is given, it is expected to be the name of a file containing a list of login sessions, in the form described in **acctcon**(8), sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file. The information in *ctmp* helps it distinguish among different login names that share the same user ID.

acctprc2

acctprc2 reads records in the form written by **acctprc1**, summarizes them by user ID and name, then writes the sorted summaries to the standard output as total accounting records.

EXAMPLES

These commands are typically used as shown below:

```
acctprc1 ctmp </var/adm/pacct | acctprc2 >ptacct
```

FILES

/etc/passwd

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctsh**(8), **cron**(8), **fwtmp**(8), **runacct**(8)

BUGS

Although it is possible to distinguish among login names that share user IDs for commands run from the command line, it is difficult to do this for those commands run by **cron**(8), for example. More precise conversion can be done by faking login sessions on the console using the **acctwtmp** program in **acct**(8).

NAME

chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, runacct, shutacct, startup, turnacct – shell procedures for accounting

SYNOPSIS

/usr/lib/acct/chargefee *login-name number*

/usr/lib/acct/ckpacct [*blocks*]

/usr/lib/acct/dodisk [**–o**] [*filename ...*]

/usr/lib/acct/lastlogin

/usr/lib/acct/monacct *number*

/usr/lib/acct/nulladm *filename*

/usr/lib/acct/prctmp *filename*

/usr/lib/acct/prdaily [**–cl**] [*mmdd*]

/usr/lib/acct/prtacct *filename* [*heading*]

/usr/lib/acct/runacct [*mmdd*] [*mmdd state*]

/usr/lib/acct/shutacct [*reason*]

/usr/lib/acct/startup

/usr/lib/acct/turnacct on | off | switch

DESCRIPTION

chargefee

chargefee can be invoked to charge a *number* of units to *login-name*. A record is written to **/var/adm/fee**, to be merged with other accounting records during the night.

ckpacct

ckpacct should be initiated by **cron**(8) every hour. It periodically checks the size of **/var/adm/pacct**. If the size exceeds *blocks*, 1000 by default, **turnacct** is called with the argument **switch**. If the number of free disk blocks in the **/usr** file system falls below 500, **ckpacct** automatically turns off the collection of process accounting records using the **off** argument to **turnacct**. When at least this number of blocks is restored, accounting is activated again. This feature is sensitive to the frequency at which **ckpacct** is executed, usually by **cron**.

dodisk

dodisk should be executed by **cron** to perform the disk accounting functions. By default, it does disk accounting on the 4.2 file systems in **/etc/fstab**. *filename*s specify the one or more filesystem names where disk accounting will be done. If *filename*s are used, disk accounting will be done on these filesystems only. They should be the special file names of mountable filesystems.

lastlogin

lastlogin is invoked by **runacct** to update **/var/adm/acct/sum/loginlog**, which shows the last date on which each person logged in. **lastlogin** deletes the entries of users no longer in **/etc/passwd** and creates new entries.

monacct

monacct should be invoked once each month or each accounting period. *number* indicates which month or period it is. If *number* is not given, it defaults to the current month (01–12). This default is useful if **monacct** is executed by **cron**(8) on the first day of each month. **monacct** creates summary files in **/var/adm/acct/fiscal** and restarts summary files in **/var/adm/acct/sum**.

nulladm

nulladm creates *filename* with mode 664 and insures that owner and group are **adm**. It is called by various accounting shell procedures.

prctmp

prctmp can be used to print the session record file with headings (normally **/var/adm/acct/nite/ctmp** created by **acctcon1** (see **acctcon**(8)). The heading specifies device, user ID, login name, prime connect time (in seconds), non-prime connect time (in seconds), session starting time (numeric) and starting date and time.

prdaily

prdaily is invoked by **runacct** to format a report of the previous day's accounting data. The report resides in **/var/adm/acct/sum/rprt***mmdd* where *mmdd* is the month and day of the report. The current daily accounting reports may be printed by typing **prdaily**. Previous days' accounting reports can be printed by using the *mmdd* option and specifying the exact report date desired. Previous daily reports are cleaned up and therefore inaccessible after each invocation of **monacct**.

prtacct

prtacct can be used to format and print any total accounting (**tacct**) file with headings. See Chapter 8 in the *System and Network Administration* manual, for an explanation of this output.

runacct

runacct performs the accumulation of connect, process, fee, and disk accounting on a daily basis. It also creates summaries of command usage. For more information, see **runacct**(8).

shutacct

shutacct should be invoked during a system shutdown (usually in **/etc/shutdown**) to turn process accounting off and append a "reason" record to **/var/adm/wtmp**. If *reason* is not specified, **shutdown** is provided as a default reason.

startup

startup should be called by **/etc/rc** to turn the accounting on whenever the system is brought up.

turnacct

turnacct is an interface to **accton** (see **acct**(8)) to turn process accounting **on** or **off**. The **switch** argument turns accounting off, moves the current **/var/adm/pacct** to the next free name in **/var/adm/pacct***incr* (where *incr* is a number starting with **1** and incrementing by one for each additional **pacct** file), then turns accounting back on again. This procedure is called by **ckpacct** and thus can be taken care of by **cron** and used to keep **pacct** to a reasonable size. This command is restricted to the super-user.

OPTIONS

dodisk

−o Do a slower version of disk accounting by login directory. *filename*s should be mount points of mounted filesystem.

prdaily

−c Prints a report of exceptional resource usage by command. This may be used on current day's accounting data only.

−l Print a report of exceptional usage by login ID for the specifed date.

FILES

/etc/fstab	list of file systems **/var/adm/fee** accumulator for fees
/var/adm/pacct	current file for per-process accounting
/var/adm/pacct*	used if pacct gets large and during execution of daily accounting procedure
/var/adm/wtmp	login/logoff summary
/usr/lib/acct/ptelus.awk	limits for exceptional usage by login id
/usr/lib/acct/ptecms.awk	limits for exceptional usage by command name
/var/adm/acct/nite	working directory
/usr/lib/acct	directory of accounting commands
/var/adm/acct/sum	summary directory, should be saved

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **cron**(8), **diskusg(8)**, **fwtmp**(8), **runacct**(8)

System and Network Administration

NAME

adbgen – generate adb script

SYNOPSIS

/usr/lib/adb/adbgen *filename*.**adb** ...

DESCRIPTION

adbgen makes it possible to write **adb**(1) scripts that do not contain hard-coded dependencies on structure member offsets. The input to **adbgen** is a file named *filename*.**adb** which contains **adbgen** header information, then a null line, then the name of a structure, and finally an **adb** script. **adbgen** only deals with one structure per file; all member names are assumed to be in this structure. The output of **adbgen** is an **adb** script in *filename*. **adbgen** operates by generating a C program which determines structure member offsets and sizes, which in turn generates the **adb** script.

The header lines, up to the null line, are copied verbatim into the generated C program. Typically these include C **#include** statements to include the header files containing the relevant structure declarations.

The **adb** script part may contain any valid **adb** commands (see **adb**(1)), and may also contain **adbgen** requests, each enclosed in {}s. Request types are:

- Print a structure member. The request form is {*member*,*format*}. *member* is a member name of the *structure* given earlier, and *format* is any valid **adb** format request. For example, to print the **p_pid** field of the *proc* structure as a decimal number, you would write **{p_pid,d}**.
- Reference a structure member. The request form is {*****member,*base*}. *member* is the member name whose value is desired, and *base* is an **adb** register name which contains the base address of the structure. For example, to get the **p_pid** field of the *proc* structure, you would get the proc structure address in an **adb** register, say <f, and write **{*p_pid,<f}**.
- Tell **adbgen** that the offset is ok. The request form is **{OFFSETOK}**. This is useful after invoking another **adb** script which moves the **adb** *dot*.
- Get the size of the *structure*. The request form is **{SIZEOF}**. **adbgen** replaces this request with the size of the structure. This is useful in incrementing a pointer to step through an array of structures.
- Get the offset to the end of the structure. The request form is **{END}**. This is useful at the end of the structure to get **adb** to align the **dot** for printing the next structure member.

adbgen keeps track of the movement of the **adb** *dot* and emits **adb** code to move forward or backward as necessary before printing any structure member in a script. **adbgen**'s model of the behavior of **adb**'s **dot** is simple: it is assumed that the first line of the script is of the form *struct_address/adb text* and that subsequent lines are of the form *+/adb text*. This causes the *adb* **dot** to move in a sane fashion. **adbgen** does not check the script to ensure that these limitations are met. **adbgen** also checks the size of the structure member against the size of the **adb** format code and warns you if they are not equal.

EXAMPLE

If there were an include file **x.h** which contained:

```
struct x {
        char    *x_cp;
        char    x_c;
        int     x_i;
};
```

Then an **adbgen** file (call it **script.adb**) to print it would be:

```
#include "x.h"
x
./"x_cp"16t"x_c"8t"x_i"n{x_cp,X}{x_c,C}{x_i,D}
```

After running **adbgen** the output file **script** would contain:

```
16t"x_c"8t"x_i"nXC+D"" ./"x_cp"16t"x_c"8t"x_i"nXC+D
```

To invoke the script you would type:

```
x$<script
```

FILES

/usr/lib/adb/*	**adb** scripts for debugging the kernel

SEE ALSO

adb(1), **kadb**(8S)

Debugging Tools

BUGS

adb syntax is ugly; there should be a higher level interface for generating scripts.

Structure members which are bit fields cannot be handled because C will not give the address of a bit field. The address is needed to determine the offset.

DIAGNOSTICS

Warnings about structure member sizes not equal to **adb** format items and complaints about badly formatted requests. The C compiler complains if you reference a structure member that does not exist. It also complains about & before array names; these complaints may be ignored.

NAME

add_client – create a diskless network bootable NFS client on a server

SYNOPSIS

/usr/etc/install/add_client [**–inpv**] [**–a** *kernel-arch*] [**–e** *exec-path*] [**–f** *share-path*] [**–h** *home-path*] [**–k** *kvm-path*] [**–m** *mail-path*] [**–r** *root-path*] [**–s** *swap-path*] [**–t** *term-type*] [**–y** *yptype*] [**–z** *swapsize*] [*client* ...]

DESCRIPTION

add_client adds an NFS client to a server. It can only be run by the super-user.

A default standard layout is used to set up the client's environment, but most pathnames can be overridden with the appropriate option, or menu field change.

Before you can add a client, you must first make sure that the Internet and Ethernet addresses for *client* are listed in the Network Interface Service (NIS) hosts database (if the server is running the NIS service), or in the server's **/etc/hosts** and **/etc/ethers** databases, respectively. If **add_client** cannot find the client entry in the hosts database it aborts the operation. If there is no client entry in the **/etc/ethers** database, **add_client** issues a warning to update this file while adding the client.

The default root and swap partitions are **/export/root/***client* and **/export/swap/***client*, respectively.

add_client updates the **/etc/bootparams** file on the server but not the bootparams database in the NIS service (if used).

If the server is not running as an NIS master, **add_client** issues a warning to indicate that the database is out of date and the NIS master should be updated.

add_client updates the server's **/etc/exports** file to allow client's root access to each client's root file system. It also exports each client's swap file accordingly. Note: the system administrator should verify that the **/etc/exports** file contains correct information, and that file systems are exported to the correct users and groups. Refer to **exportfs**(8) for details on exporting file systems.

If the **–i** or **–p** option is not specified, at least one *client* argument must be supplied on the command line.

OPTIONS

–i Interactive. Bring up a full-screen menu interface to **add_client**.

–n Print the working parameters and exit without doing anything. This is used to verify what parameters **add_client** will use before actually doing anything.

–p Display a short version of all client information, If *client*s are specified on the command line, only display information for those clients. When combined with the **–v** option, a long version of client information is displayed.

–v Verbose. Report information about the client as steps are performed.

–a *kernel-arch* Specify the client kernel architecture (for instance, **sun3**, **sun4**, **sun4c**...). **add_client** prompts for the kernel architecture when unable to determine the correct value.

–e *exec-path* Set the pathname of the directory in which the executables for the architecture specified by **–a**. The client mounts **/export/exec/***arch.rel* as **/usr**. See WARNINGS.

–f *share-path* Set the pathname of the share directory, which is normally a link to **/usr/share**.

–h *home-path* Set the pathname of the directory for the client's home. The default is **/home/***server-name*.

–k *kvm-path* Set the pathname of the directory containing the client's kernel executables. See WARNINGS.

–m *mail-path* Set the pathname of the client's mail directory. The default is **/var/spool/mail**.

–r *root-path* Set the pathname of parent directory for client root directories; *root/client* is the pathname of the client's root directory. The default is **/export/root/***client-name*.

-s *swap-path* Set the pathname of parent directory for client swap files; *swap/client* is the pathname of the client's swap file. The default is **/export/swap/***client-name*.

-t *term-type* Set the terminal type of the client's console.

-y *yptype* Indicate the type of NIS server or if client is to be an NIS client; it can be **client** or **none**. The **none** argument results in the NIS service being disabled on the client. The default is **client**.

-z *swapsize* Reserve *swapsize* bytes for the client's swap file. **swapsize** can be flagged as kilobytes, blocks, or megabytes, with the **k**, **b**, or **m** suffixes, respectively. The default is 16Mb, and bytes are used when no units are specified.

FILES

/etc/bootparams
/etc/ethers
/etc/exports
/etc/hosts
/export/exec/proto.root.*release*
architecture independent base for the client root file system
/tftpboot.*client-ipaddr* link to **/tftpboot/boot.***arch*

SEE ALSO

add_services(8), **bootparamd(8)**, **exportfs**(8), **ndbootd**(8C), **rm_client**(8), **suninstall**(8)

Installing SunOS 4.1

DIAGNOSTICS

add_client: must be super-user
You must be root to use **add_client**.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

WARNINGS

The **-e** *exec-path* and the **-k** *kvm-path* options should not be used since the correct paths are determined when the adding the client's architecture service. See **add_services**(8).

NAME

add_services – provide software installation services for any architecture

SYNOPSIS

/usr/etc/install/add_services

DESCRIPTION

add_services is a menu-based program to setup a system as a server and/or to add additional software categories or other architecture releases. It is used to provide support to diskless clients, dataless clients, or just to act as a file server. **add_services** can only be run by the super-user.

add_services updates the **/etc/exports** file (see **exports**(5) and **exportfs**(8)) to export the necessary file systems to become a file server. After running **add_services**, the system administrator should verify this file to make sure that the new services have been exported to the correct groups.

FILES

/etc/hosts	hosts database, host must be in this database or in the Network Interface Service (NIS) hosts map
/etc/exports	database of exported file systems, service related directories must be exported
/tftpboot	add_services sets up this directory in order to provide boot service to clients

SEE ALSO

exports(5), **add_client**(8), **exportfs**(8), **rm_client**(8), **suninstall**(8)

Installing SunOS 4.1

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

adduser – procedure for adding new users

DESCRIPTION

To add an account for a new user, the system administrator (or super-user):

- Create an entry for the new user in the system password files.
- Create a home directory for the user, and change ownership so the new user owns that directory.
- Optionally set up skeletal dot files for the new user (**.cshrc**, **.login**, **.profile**...).
- If the account is on a system running the Network Interface Service (NIS), take additional measures.

USAGE

Making an Entry in the Password File

To add an entry for the new login name on a local host, first edit the **/etc/passwd** file — inserting a line for the new user. This must be done with the password file locked, for instance, by using **vipw**(8), and the insertion must be made above the line containing the string:

+::0:0:::

This line indicates that additional accounts can be found in the NIS service.

To add an entry for the new login name into the NIS service, add an identical line to the file **/etc/passwd** on the NIS master server, and run **make**(1) in the directory **/var/yp** (see **ypmake**(8) for details) to propagate the change.

The new user is assigned a group and user ID number (GID and UID respectively). UIDs should be unique for each user and consistent across the NFS domain, since they control access to files. GIDs need not be unique. Typically, users working on similar projects will assigned to the same group. The system staff is group 10 for historical reasons, and the super-user is in this group.

An entry for a new user **francine** would look like this:

francine::235:20:& Featherstonehaugh:/usr/francine:/bin/csh

Fields in each password-file entry are delimited by colons, and have the following meanings:

- Login name (**francine**). The login name is limited to eight characters in length.
- Encrypted password or the string *##name* if encrypted passwords are stored in the password adjunct file. Typically, if passwords are to be stored in the main password file, this field is left empty, so no password is needed when the user first logs in. If security demands a password, it should be assigned by running **passwd**(1) immediately after exiting the editor. The number of significant characters in a password is eight. (See **passwd**(1).)
- User ID. The UID is a number which identifies that user uniquely in the system. Files owned by the user have this number stored in their data blocks, and commands such as **ls (1V)** (see **ls**(1V)), use it to look up the owner's login name. For this reason, you cannot randomly change this number. See **passwd**(5) for more information.
- Group ID. The GID number identifies the group to which the user belongs by default (although the user may belong to additional groups as well). All files that the user creates have this number stored in their data blocks, and commands such as **ls**(1V) (see **ls**(1V), use it to look up the group name. Group names and assignments are listed in the file **/etc/group** (which is described in **group**(5)) or in the NIS group map.
- This field is called the GCOS field (from earlier implementation of the operating system) and is traditionally used to hold the user's full name. Some installations have other information encoded in this field. From this information we can tell that Francine's real name is 'Francine Featherstonehaugh'. The & in the entry is shorthand for the user's login name.

- User's home directory. This is the directory in which that user is "positioned" when they log in.
- Initial shell which this user will see on login. If this field is empty, **sh**(1) is used as the initial shell.

An entry for a new user **francine** would look like this:

francine:::::lo:ad,+dw

Fields in each password adjunct file entry are delimited by colons, and have the following meanings:

- Login name (**francine**). This name must match the login name in the password file.
- Encrypted password. Typically, this field is left empty when adding the line using the editor. **passwd**(1) should be run immediately after exiting the editor.
- The next three fields are the minimum label, the maximum label, and the default label. These fields should be left empty, since they are reserved for future use.
- The next two fields are for the always-audit flags and the never-audit flags. Always-audit flags specify which events are guaranteed to be audited for that user. Never-audit flags specify which events are guaranteed not to be audited for that user. For a description of audit flags, see **audit_data**(5).

Making a Home Directory

As shown in the password file entry above, the name of Francine's home directory is to be **/usr/francine**. This directory must be created using **mkdir**(1), and Francine must be given ownership of it using **chown**(8), in order for her profile files to be read and executed, and to have control over access to it by other users:

```
example# mkdir /usr/francine
example# /usr/etc/chown francine /usr/francine
```

If running under NFS, the **mkdir**(1) and **chown**(8) commands must be performed on the NFS server.

Setting Up Skeletal Profile Files

New users often need assistance in setting up their profile files to initialize the terminal properly, configure their search path, and perform other desired functions at startup. Providing them with skeletal profile files saves time and interruptions for both the new user and the system administrator.

Such files as **.profile** (if they use **/usr/bin/sh** as the shell), or **.cshrc** and **.login** (if they use **/usr/bin/csh** as the shell), can include commands that are performed automatically at each login, or whenever a shell is invoked, such as **tset**(1). The ownership of these files must be changed to belong to the new user, either by running **su**(1V) before making copies, or by using **chown**(8).

FILES

/etc/passwd	password file
/etc/security/passwd.adjunct	
/etc/group	group file
/etc/yp/src/passwd	
~/.cshrc	
~/.login	
~/.profile	

SEE ALSO

csh(1), **ls**(1V), **make**(1), **mkdir**(1), **passwd**(1), **sh**(1), **su**(1V), **tset**(1), **audit**(2), **audit_control**(5), **audit_data**(5), **passwd.adjunct**(5), **group**(5), **passwd**(5), **passwd.adjunct**(5) **audit**(8), **auditd**(8), **chown**(8), **vipw**(8), **ypmake**(8)

System and Network Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

adv – advertise a directory for remote access with RFS

SYNOPSIS

adv
adv [**–r**] [**–d** *description*] *resource pathname* [*clients* . . .]
adv –m *resource* **–d** *description* | [*clients*. . .]
adv –m *resource* [**–d** *description*] | *clients* . . .

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

adv makes a resource from one system available for use on other systems. The machine that advertises the resource is called the server, while systems that mount and use the resource are **clients**. See **mount**(8). *resource* represents a directory, which could contain files, subdirectories, named pipes and devices.

Remote File Sharing (RFS) must be running before **adv** can be used to advertise or modify a resource entry.

When used with no options, **adv** displays all local resources that have been advertised; this includes the resource name, the pathname, the description, the read-write status, and the list of authorized clients. The resource field has a fixed length of 14 characters; all others are of variable length. Fields are separated by two SPACE characters and double quotes (") surround the description.

This command may be used without options by any user; otherwise it is restricted to the super-user.

There are three ways **adv** is used:

- To print a list of all locally-advertised resources, as shown by the first synopsis.
- To advertise the directory *pathname* under the name *resource* so it is available to RFS *clients*, as shown by the second synopsis.
- To modify *client* and *description* fields for currently advertised resources, as shown by the third and fourth synopses.

If any of the following are true, an error message will be sent to standard error.

- The network is not up and running.
- *pathname* is not a directory.
- *pathname* is not on a file system mounted locally.
- There is at least one entry in the *clients* field but none are syntactically valid.

OPTIONS

–r Restrict access to the resource to a read-only basis. The default is read-write access.

–d *description* Provide brief textual information about the advertised resource. *description* is a single argument surrounded by double quotes ("*argument*") and has a maximum length of 32 characters.

–m *resource* Modify information for a resource that has already been advertised. The resource is identified by a *resource* name. Only the *clients* and *description* fields can be modified. To change the *pathname*, *resource* name, or read/write permissions, you must unadvertise and re-advertise the resource.

resource This is the symbolic name used by the server and all authorized clients to identify the resource. It is limited to a maximum of 14 characters and must be different from every other resource name in the domain. All characters must be printable ASCII characters, but must not include '.' (periods), '/' (slashes), or white space.

pathname This is the local pathname of the advertised resource. It is limited to a maximum of 64 characters. This pathname cannot be the mount point of a remote resource and it can only be advertised under one resource name.

clients These are the names of all clients that are authorized to remotely mount the resource. The default is that all machines that can connect to the server are authorized to access the resource. Valid input is of the form *nodename*, *domain.nodename*, *domain.*, or an alias that represents a list of client names. A domain name must be followed by a '.' to distinguish it from a host name. The aliases are defined in **/etc/host.alias** and must conform to the alias capability in **mail**(1).

EXAMPLES

The following example displays the local resources that have been advertised:

```
example% adv
LOCAL_SUN3    /export/local/sun3 "" read-only unrestricted
LOCAL_SUN4    /export/local/sun4 "" read-only unrestricted
LOCAL_SHARE   /export/local/share "" read-only unrestricted
```

EXIT STATUS

If there is at least one syntactically valid entry in the *clients* field, a warning will be issued for each invalid entry and the command will return a successful exit status. A non-zero exit status will be returned if the command fails.

FILES

/etc/host.alias

SEE ALSO

mount(8), **rfstart**(8), **unadv**(8)

NAME

arp – address resolution display and control

SYNOPSIS

arp *hostname*

arp **–a** [*vmunix* [*kmem*]]

arp **–d** *hostname*

arp **–s** *hostname ether_address* [**temp**] [**pub**] [**trail**]

arp **–f** *filename*

DESCRIPTION

The **arp** program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (**arp**(4P)).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation.

OPTIONS

–a Display all of the current ARP entries by reading the table from the file *kmem* (default **/dev/kmem**) based on the kernel file *vmunix* (default **/vmunix**).

–d Delete an entry for the host called *hostname*. This option may only be used by the super-user.

–s Create an ARP entry for the host called *hostname* with the Ethernet address *ether_address*. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word **temp** is given in the command. If the word **pub** is given, the entry will be published, for instance, this system will respond to ARP requests for *hostname* even though the hostname is not its own. The word **trail** indicates that trailer encapsulations may be sent to this host.

–f Read the file named *filename* and set multiple entries in the ARP tables. Entries in the file should be of the form

hostname ether_address [**temp**] [**pub**] [**trail**]

with argument meanings as given above.

SEE ALSO

arp(4P), **ifconfig**(8C)

NAME

audit – audit trail maintenance

SYNOPSIS

audit [**–n** | **–s** | **–t**]
audit **–d** *username*
audit **–u** *username audit_event_state*

AVAILABILITY

This program is available with the *Security* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

The **audit** command is the general administrator's interface to kernel auditing. The process audit state for a user can be temporarily or permanently altered. The audit daemon may be notified to read the contents of the **audit_control** file and re-initialize the current audit directory to the first directory listed in the **audit_control** file, or to open a new audit file in the current audit directory specified in the **audit_control** file as last read by the audit daemon. Auditing may also be terminated/disabled.

OPTIONS

–n Signal audit daemon to close the current audit file and open a new audit file in the current audit directory.

–s Signal audit daemon to read audit control file. The audit daemon stores the information internally.

–t Signal audit daemon to disable auditing and die.

–d *username*
Change the process audit state of all processes owned by *username*. This new process audit state is constructed from the system and user audit values as specified in the **audit_control** and **passwd.adjunct** files respectively.

–u *username audit_event_state*
Set the process audit state from *audit_event_state* for all current processes owned by *username*. See **audit_control**(5) for the format of the system audit value. The process audit state is one argument. Enclose the audit event state in quotes, or do not use SPACE characters in the process audit state specification. A new login session reconstructs the process audit state from the audit flags in the **audit_control** and **passwd.adjunct** files.

SEE ALSO

audit(2), **setuseraudit**(2), **getauditflags**(3), **getfauditflags**(3), **audit_control**(5), **passwd.adjunct**(5)

NAME

auditd – audit daemon

SYNOPSIS

/usr/etc/auditd

DESCRIPTION

The audit daemon controls the generation and location of audit trail files. If the function **issecure**(3) returns false, the only action that **auditd** takes is to disable the auditing system; otherwise, auditing is set up and started. If auditing is desired, **auditd** reads the **audit_control**(5) file to get a list of directories into which audit files can be written and the percentage limit for how much space to reserve on each filesystem before changing to the next directory.

If **auditd** receives the signal **SIGUSR1**, the current audit file is closed and another is opened. If **SIGHUP** is received, the current audit trail is closed, the **audit_control** file reread, and a new trail is opened. If **SIGTERM** is received, the audit trail is closed and auditing is terminated. The program **audit**(8) sends these signals and is recommended for this purpose.

Each time the audit daemon opens a new audit trail file, it updates the file **audit_data**(5) to include the correct name.

Auditing Conditions

The audit daemon invokes the program **audit_warn**(8) under the following conditions with the indicated options:

audit_warn soft *pathname*

The file system upon which *pathname* resides has exceeded the minimum free space limit defined in **audit_control**(5). A new audit trail has been opened on another file system.

audit_warn allsoft

All available file systems have been filled beyond the minimum free space limit. A new audit trail has been opened anyway.

audit_warn hard *pathname*

The file system upon which *pathname* resides has filled or for some reason become unavailable. A new audit trail has been opened on another file system.

audit_warn allhard *count*

All available file systems have been filled or for some reason become unavailable. The audit daemon will repeat this call to **audit_warn** every twenty seconds until space becomes available. *count* is the number of times that **audit_warn** has been called since the problem arose.

audit_warn ebusy

There is already an audit daemon running.

audit_warn tmpfile

The file **/etc/security/audit/audit_tmp** exists, indicating a fatal error.

audit_warn nostart

The internal system audit condition is **AUC_FCHDONE**. Auditing cannot be started without rebooting the system.

audit_warn auditoff

The internal system audit condition has been changed to not be **AUC_AUDITING** by someone other than the audit daemon. This causes the audit daemon to exit.

audit_warn postsigterm

An error occurred during the orderly shutdown of the auditing system.

audit_warn getacdir

There is a problem getting the directory list from **/etc/security/audit/audit_control**.

The audit daemon will hang in a sleep loop until this file is fixed.

FILES

/etc/security/audit/audit_control
/etc/security/audit/audit_data

SEE ALSO

auditsvc(2), **audit_control**(5), **audit.log**(5), **audit**(8), **audit_warn**(8)

NAME

audit_warn – audit daemon warning script

SYNOPSIS

/usr/etc/audit_warn [*option* [*arguments*]]

DESCRIPTION

The **audit_warn** script processes warning or error messages from the audit daemon. When a problem is encountered, the audit daemon, **auditd**(8) calls **audit_warn** with the appropriate arguments. The *option* argument specifies the error type.

The system administrator can specify a list of mail recpients using the script's **RECIPIENTS** variable. The default recipient is root.

OPTIONS

soft *filename*
indicates that the soft limit for *filename* has been exceeded. The default action for this option is to send mail to the system administrator.

allsoft indicates that the soft limit for all filesystems has been exceeded. The default action for this option is to send mail to the system administrator.

hard *filename*
indicates that the hard limit for the file has been exceeded. The default action for this option is to send mail to the system administrator.

allhard *count*
indicates that the hard limit for all filesystems has been exceeded *count* times. The default action for this option is to send mail to the system administrator only if the *count* is **1**, and to send a message to console every time. It is recommended that mail *not* be send every time.

ebusy indicates that the audit daemon is already running. The default action for this option is to send mail to the system administrator.

tmpfile indicates that the temporary audit file already exists indicating a fatal error. The default action for this option is to send mail to the system administrator.

nostart indicates that auditing cannot be started because the system audit state is **AUC_FCHDONE**. The default action for this option is to send mail to the system administrator. Some system administrators may prefer to have the script reboot the system at this point.

auditoff
indicates that someone other than the audit daemon changed the system audit state to something other than **AUC_AUDITING**. The audit daemon will have exited in this case. The default action for this option is to send mail to the system administrator.

postsigterm
indicates that an error occurred during the orderly shutdown of the audit daemon. The default action for this option is to send mail to the system administrator.

getacdir
indicates that there is a problem getting the directory list from: **/etc/security/audit/audit_control.**

The audit daemon will hang in a sleep loop until the file is fixed.

SEE ALSO

audit.log(5), **audit_control**(5), **audit**(8), **auditd**(8)

NAME

automount – automatically mount NFS file systems

SYNOPSIS

automount [**–mnTv**] [**–D** *name= value*] [**–f** *master-file*] [**–M** *mount-directory*] [**–tl** *duration*] [**–tm** *interval*] [**–tw** *interval*] [*directory map* [*–mount-options*]] ...

DESCRIPTION

automount is a daemon that automatically and transparently mounts an NFS file system as needed. It monitors attempts to access directories that are associated with an **automount** map, along with any directories or files that reside under them. When a file is to be accessed, the daemon mounts the appropriate NFS file system. You can assign a map to a directory using an entry in a direct **automount** map, or by specifying an indirect map on the command line.

The **automount** daemon appears to be an NFS server to the kernel. **automount** uses the map to locate an appropriate NFS file server, exported file system, and mount options. It then mounts the file system in a temporary location, and creates a symbolic link to the temporary location. If the file system is not accessed within an appropriate interval (five minutes by default), the daemon unmounts the file system and removes the symbolic link. If the indicated directory has not already been created, the daemon creates it, and then removes it upon exiting.

Since the name-to-location binding is dynamic, updates to an **automount** map are transparent to the user. This obviates the need to "pre-mount" shared file systems for applications that have "hard coded" references to files.

If the *directory* argument is a pathname, the *map* argument must be an *indirect* map. In an indirect map the key for each entry is a simple name that represents a symbolic link within *directory* to an NFS mount point.

If the *directory* argument is '/–', the map that follows must be a *direct* map. A direct map is not associated with a single directory. Instead, the key for each entry is a full pathname that will itself appear to be a symbolic link to an NFS mount point.

A map can be a file or a Network Interface Service (NIS) map; if a file, the *map* argument must be a full pathname.

The *–mount-options* argument, when supplied, is a comma-separated list of **mount**(8) options, preceded by a '–'. If these options are supplied, they become the default mount options for all entries in the map. Mount options provided within a map entry override these defaults.

OPTIONS

–m Suppress initialization of *directory-map* pairs listed in the **auto.master** NIS database.

–n Disable dynamic mounts. With this option, references through the **automount** daemon only succeed when the target filesystem has been previously mounted. This can be used to prevent NFS servers from cross-mounting each other.

–T Trace. Expand each NFS call and display it on the standard output.

–v Verbose. Log status and/or warning messages to the console.

–D *envar=value*
Assign *value* to the indicated **automount** (environment) variable.

–f *master-file*
Read a local file for initialization, ahead of the **auto.master** NIS map.

–M *mount-directory*
Mount temporary file systems in the named directory, instead of **/tmp_mnt**.

–tl *duration*
Specify a *duration*, in seconds, that a file system is to remain mounted when not in use. The default is 5 minutes.

–tm *interval*
Specify an *interval*, in seconds, between attempts to mount a filesystem. The default is 30 seconds.

–tw *interval*
Specify an *interval*, in seconds, between attempts to unmount filesystems that have exceeded their cached times. The default is 1 minute.

ENVIRONMENT

Environment variables can be used within an **automount** map. For instance, if **$HOME** appeared within a map, **automount** would expand it to its current value for the **HOME** variable. Environment variables are expanded only for the automounter's environment — not for the environment of a user using the automounter's services.

The special reference to **$ARCH** expands to the output of **arch (1).** This can be useful in creating a map entry for mounting executables using a server's export pathname that varies according to the architecture of the client reading the map.

If a reference needs to be protected from affixed characters, you can surround the variable name with curly braces.

USAGE

Map Entry Format

A simple map entry (mapping) takes the form:

key [*–mount-options*] *location* ...

where *key* is the full pathname of the directory to mount when used in a direct map, or simple name in an indirect map. *mount-options* is a comma-separated list of **mount** options, and *location* specifies a remote filesystem from which the directory may be mounted. In the simple case, *location* takes the form:

hostname:*pathname*

Replicated Filesystems

Multiple *location* fields can be specified for replicated read-only filesystems, in which case **automount** sends multiple **mount** requests; **automount** mounts the file system from the first host that replies to the **mount** request. This request is first made to the local net or subnet. If there is no response, any connected server may respond. Since **automount** does not monitor the status of the server while the filesystem is mounted it will not use another location in the list if the currently mounted server crashes. This support for replicated filesystems is available only at mount time.

If each *location* in the list shares the same *pathname* then a single *location* may be used with a comma-separated list of hostnames.

hostname,*hostname* ... : *pathname*

Sharing Mounts

If *location* is specified in the form:

hostname:*pathname*:*subdir*

hostname is the name of the server from which to mount the file system, *pathname* is the pathname of the directory to mount, and *subdir*, when supplied, is the name of a subdirectory to which the symbolic link is made. This can be used to prevent duplicate mounts when multiple directories in the same remote file system may be accessed. With a map for **/home** such as:

able **homeboy:/home/homeboy:able**
baker **homeboy:/home/homeboy:baker**

and a user attempting to access a file in **/home/able**, **automount** mounts **homeboy:/home/homeboy**, but creates a symbolic link called **/home/able** to the **able** subdirectory in the temporarily-mounted filesystem. If a user immediately tries to access a file in **/home/baker**, **automount** needs only to create a symbolic link that points to the **baker** subdirectory; **/home/homeboy** is already mounted.

With the following map:

able homeboy:/home/homeboy/able
baker homeboy:/home/homeboy/baker

automount would have to mount the filesystem twice.

Comments and Quoting

A mapping can be continued across input lines by escaping the NEWLINE with a backslash. Comments begin with a # and end at the subsequent NEWLINE.

Characters that have special significance to the **automount** map parser may be protected either with double quotes (") or by escaping with a backslash (\). Pathnames with embedded whitespace, colons (:) or dollar ($) should be protected.

Directory Pattern Matching

The '**&**' character is expanded to the value of the *key* field for the entry in which it occurs. In this case:

able homeboy:/home/homeboy:&

the & expands to **able**.

The '*' character, when supplied as the *key* field, is recognized as the catch-all entry. Such an entry will be used if any previous entry has not successfully matched the key being searched for. For instance, if the following entry appeared in the indirect map for **/home**:

*** &:/home/&**

this would allow automatic mounts in **/home** of any remote file system whose location could be specified as:

hostname:**/home/***hostname*

Multiple Mounts

A multiple mount entry takes the form:

key [/[*mountpoint* [*−mount-options*] *location* ...] ...

The initial / within the '/[*mountpoint*]' is required; the optional *mountpoint* is taken as a pathname relative to the destination of the symbolic link for *key*. If *mountpoint* is omitted in the first occurrence, a *mountpoint* of / is implied.

Given the direct map entry:

```
/arch/src  \
        /          −ro,intr    arch:/arch/src          alt:/arch/src  \
        /1.0       −ro,intr    alt:/arch/src/1.0       arch:/arch/src/1.0  \
        /1.0/man   −ro,intr    arch:/arch/src/1.0/man  alt:/arch/src/1.0/man
```

automount would automatically mount **/arch/src**, **/arch/src/1.0** and **/arch/src/1.0/man**, as needed, from either **arch** or **alt**, whichever host responded first. If the mounts are hierarchically related mounts closer to the root must appear before submounts. All the mounts of a multiple mount entry will occur together and will be unmounted together. This is important if the filesystems reference each other with relative symbolic links. Multiple mount entries can be used both in direct maps and in indirect maps.

Included Maps

The contents of another map can be included within a map with an entry of the form:

+mapname

mapname can either be a filename, or the name of an NIS map, or one of the special maps described below. If the key being searched for is not located in an included map, the search continues with the next entry.

Special Maps

There are two special maps currently available: **–hosts**, and **–null**. The **–hosts** map uses the NIS **hosts.byname** map to locate a remote host when the hostname is specified. This map specifies mounts of all exported file systems from any host. For instance, if the following **automount** command is already in effect:

automount /net –hosts

then a reference to **/net/hermes/usr** would initiate an automatic mount of all file systems from **hermes** that **automount** can mount; references to a directory under **/net/hermes** will refer to the corresponding directory relative to **hermes** root.

The **–null** map, when indicated on the command line, cancels any subsequent map for the directory indicated. It can be used to cancel a map given in **auto.master** or for a mount point specified as an entry in a direct map.

Configuration and the auto.master Map

automount normally consults the **auto.master** NIS configuration map for a list of initial **automount** maps, and sets up automatic mounts for them in addition to those given on the command line. If there are duplications, the command-line arguments take precedence over a local **–f** master map and they both take precedence over an NIS **auto.master** map. This configuration database contains arguments to the **automount** command, rather than mappings; unless **–f** is in effect, **automount** does *not* look for an **auto.master** file on the local host.

Maps given on the command line, or those given in a local **auto.master** file specified with **–f** override those in the NIS **auto.master** map. For instance, given the command:

automount –f /etc/auto.master /home –null /– /etc/auto.direct

and a file named **/etc/auto.master** that contains:

/home auto.home

automount would ignore /home entry in /etc/auto.master.

FILES

/tmp_mnt	directory under which filesystems are dynamically mounted

SEE ALSO

df(1V), **ls**(1V), **stat**(2V), **passwd**(5), **mount**(8)

System and Network Administration

NOTES

The **–hosts** map must mount all the exported filesystems from a server. If frequent access to just a single filesystem is required it is more efficient to access the filesystem with a map entry that is tailored to mount just the filesystem of interest.

When it receives signal number 1, **SIGHUP**, **automount** rereads the **/etc/mtab** file to update its internal record of currently-mounted file systems. If a file system mounted with **automount** is unmounted by a **umount** command, **automount** should be forced to reread the file.

An **ls**(1V) listing of the entries in the directory for an indirect map shows only the symbolic links for currently mounted filesystems. This restriction is intended to avoid unnecessary mounts as a side effect of programs that read the directory and **stat**(2V) each of the names.

Mount points for a single automounter must not be hierarchically related. **automount** will not allow an automount mount point to be created within an automounted filesystem.

automount must not be terminated with the **SIGKILL** signal (**kill –9**). Without an opportunity to unmount itself, the **automount** mount points will appear to the kernel to belong to a non-responding NFS server. The recommended way to terminate **automount** services is to send a **SIGTERM** (**kill –15**) signal to the daemon. This allows the automounter to catch the signal and unmount not only its daemon but also any mounts in **/tmp_mnt**. Mounts in **/tmp_mnt** that are busy will not be unmounted.

Since each direct map entry results in a separate mount for the mount daemon such maps should be kept short. Entries added to a direct map will have no effect until the automounter is restarted.

Entries in both direct and indirect maps can be modified at any time. The new information will be used when **automount** next uses the map entry to do a mount. **automount** does not cache map entries.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

BUGS

The **bg** mount option is not recognized by the automounter.

Since **automount** is single-threaded, any request that is delayed by a slow or non-responding NFS server will delay all subsequent automatic mount requests until it completes.

Programs that read **/etc/mtab** and then touch files that reside under automatic mount points will introduce further entries to the file.

Automatically-mounted file systems are mounted with type **ignore**; they do not appear in the output of either **mount**(8), or **df**(1V).

NAME

boot – start the system kernel, or a standalone program

SYNOPSIS

>b [*device* [(*c*,*u*,*p*)]] [*filename*] [**–av**] *boot-flags*
>b?
>b!

DESCRIPTION

The boot program is started by the PROM monitor and loads the kernel, or another executable program, into memory.

The form **b?** displays all boot devices and their device arguments.

The form **b!** boots, but does not perform a RESET.

USAGE

Booting Standalone

When booting standalone, the boot program (**/boot**) is brought in by the PROM from the file system. This program contains drivers for all devices.

Booting a Sun-3 System Over the Network

When booting over the network, the Sun-3 system PROM obtains a version of the boot program from a server using the Trivial File Transfer Protocol (TFTP). The client broadcasts a RARP request containing its Ethernet address. A server responds with the client's Internet address. The client then sends a TFTP request for its boot program to that server (or if that fails, it broadcasts the request). The filename requested (unqualified — not a pathname) is the hexadecimal, uppercase representation of the client's Internet address, for example:

Using IP Address 192.9.1.17 = C0090111

When the Sun server receives the request, it looks in the directory **/tftpboot** for *filename*. That file is typically a symbolic link to the client's boot program, normally **boot.sun3** in the same directory. The server invokes the TFTP server, **tftpd**(8C), to transfer the file to the client.

When the file is successfully read in by the client, the boot program jumps to the load-point and loads **vmunix** (or a standalone program). In order to do this, the boot program makes a broadcast RARP request to find the client's IP address, and then makes a second broadcast request to a **bootparamd**(8) bootparams daemon, for information necessary to boot the client. The bootparams daemon obtains this information either from a local **/etc/bootparams** database file, or from a Network Interface Service (NIS) map. The boot program sends two requests to the bootparams daemon, the first, **whoami**, to obtain its hostname, and the second, **getfile**, to obtain the name of the client's server and the pathname of the client's root partition.

The boot program then performs a **mount**(8) operation to mount the client's root partition, after which it can read in and execute any program within that partition by pathname (including a symbolic link to another file within that same partition). Typically, it reads in the file **/vmunix**. If the program is not read in successfully, **boot** responds with a short diagnostic message.

Booting a Sun-2, Sun-4, or Sun386i System Over the Network

Sun-2, Sun-4 and Sun386i systems boot over the network in a similar fashion. However, the filename requested from a server must have a suffix that reflects the system architecture of the machine being booted. For these systems, the requested filename has the form:

ip-address.arch

where *ip-address* is the machine's Internet Protocol (IP) address in hex, and *arch* is a suffix representing its architecture. (Only Sun-3 systems may omit the *arch* suffix.) These filenames are restricted to 14 characters for compatibility with System V and other operating systems. Therefore, the architecture suffix is limited to 5 characters; it must be in upper case. At present, the following suffixes are recognized: **SUN2** for Sun-2 system, **SUN3** for Sun-3 system, **SUN4** for Sun-4 system, **S386** for Sun386i system, and **PCNFS** for PC-NFS. That file is typically a symbolic link to the client's boot program, normally **boot.sun2** in the same directory for a Sun-2 system, **boot.sun3** in the same directory for a Sun-3 system, or **boot.sun4** in the same directory for a Sun-4 system.

Note: a Sun-2 system boots from its server using one extra step. It broadcasts an ND request which is intercepted by the user-level **ndbootd (8C)** (see **ndbootd**(8C) server. This server sends back a standalone program that carries out the same TFTP request sequence as is done for all the other systems.

System Startup

Once the system is loaded and running, the kernel performs some internal housekeeping, configures its device drivers, and allocates its internal tables and buffers. The kernel then starts process number 1 to run **init**(8), which performs file system housekeeping, starts system daemons, initializes the system console, and begins multiuser operation. Some of these activities are omitted when **init** is invoked with certain *boot-flags*. These are typically entered as arguments to the boot command, and passed along by the kernel to **init.**

OPTIONS

device One of:

ie	Intel Ethernet
ec	3Com Ethernet
le	Lance Ethernet
sd	SCSI disk
st	SCSI 1/4" tape
mt	Tape Master 9-track 1/2" tape
xt	Xylogics 1/2" tape
xy	Xylogics 440/450/451 disk

c Controller number, **0** if there is only one controller for the indicated type of device.

u Unit number, **0** if only there is only one driver.

filename Name of a standalone program in the selected partition, such as **stand/diag** or **vmunix.** Note: *filename* is relative to the root of the selected device and partition. It never begins with '/' (slash). If *filename* is not given, the boot program uses a default value (normally **vmunix**). This is stored in the **vmunix** variable in the **boot** executable file supplied by Sun, but can be patched to indicate another standalone program loaded using **adb**(1).

–a Prompt interactively for the device and name of the file to boot. For more information on how to boot from a specific device, refer to *Installing SunOS 4.1.*

–v Verbose. Print more detailed information to assist in diagnosing diskless booting problems.

boot-flags The boot program passes all *boot-flags* to the kernel or standalone program. They are typically arguments to that program or, as with those listed below, arguments to programs that it invokes.

–b Pass the **–b** flag through the kernel to **init**(8) so as to skip execution of the **/etc/rc.boot** script.

–h Halt after loading the system.

–s Pass the **–s** flag through the kernel to **init**(8) for single-user operation.

–i *initname*
Pass the **–i** *initname* to the kernel to tell it to run *initname* as the first program rather than the default **/sbin/init.**

FILES

/boot	standalone boot program
/tftpboot/*address*	symbolic link to the boot program for the client whose Internet address, in uppercase hexadecimal, is *address*
/tftpboot/boot.sun3	Sun-3 first stage boot program
/tftpboot/boot.sun4	Sun-4 first stage boot program
/usr/etc/in.tftpd	TFTP server
/usr/mdec/installboot	program to install boot blocks from a remote host
/vmunix	kernel file that is booted by default
/etc/bootparams	file defining root and swap paths for clients

SEE ALSO

adb(1), **tftp**(1C) **bootparamd**(8), **init**(8), **kadb**(8S), **monitor**(8S), **mount**(8), **ndbootd**(8C), **rc**(8), **reboot**(8), **tftpd**(8C)

Installing SunOS 4.1
System and Network Administration

BUGS

On Sun-2 systems, the PROM passes in the default name **vmunix**, overriding the the boot program's patchable default.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

bootparamd – boot parameter server

SYNOPSIS

/usr/etc/rpc.bootparamd [**–d**]

DESCRIPTION

bootparamd is a server process that provides information to diskless clients necessary for booting. It first consults the local **/etc/bootparams** file for a client entry. If the local bootparams file does not exist, **bootparamd** consults the corresponding Network Interface Service (NIS) map.

bootparamd can be invoked either by **inetd**(8C) or by the user.

OPTIONS

–d Display the debugging information.

FILES

/etc/bootparams

SEE ALSO

inetd(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

C2conv, C2unconv – convert system to or from C2 security

SYNOPSIS

C2conv

C2unconv

AVAILABILITY

This program is available with the *Security* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

C2conv converts a standard SunOS system to operate with C2-level security.

The program prompts for information regarding the Secure NFS option, client systems (if the system is an NFS server for diskless clients), audit devices (if it is an audit file server), and names of file systems (if there is a remote audit server). The program also requests certain information for the **audit_control**(5) file; default values may be used for audit flags and for the "minfree" value. Finally, it requests the mail address to be used (by **mail**(1)) when C2 administrative tasks are required. The default address is **root** for the host being converted.

Once it has this information, **C2conv** uses it to set up the necessary files for a C2 secure system, reporting on its progress as it proceeds.

C2unconv backs out the changes made to **/etc/passwd** and **/etc/group**. It does not back out changes to other files.

FILES

/etc/passwd
/etc/group
/etc/fstab

SEE ALSO

audit_control(5)

NAME

captoinfo – convert a termcap description into a terminfo description

SYNOPSIS

captoinfo [**–v** ...] [**–V**] [**–1**] [**–w** *width*] *filename*...

SYNOPSIS

/usr/5bin/captoinfo [**–v** ...] [**–V**] [**–1**] [**–w** *width*] *filename*...

AVAILABILITY

The System V version of this command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

captoinfo converts the **termcap**(5) terminal description entries given in *filename* into **terminfo**(5V) source entries, and writes them to the standard output along with any comments found in that file. A description that is expressed as relative to another description (as specified in the **termcap** *tc*= capability) is reduced to the minimum superset before being written.

If no *filename* is given, then the environment variable **TERMCAP** is used for the filename or entry. If **TERMCAP** is a full pathname to a file, only the terminal-name is specified in the environment variable **TERM** is extracted from that file. If that environment variable is not set, then the file **/etc/termcap** is read.

OPTIONS

–v Verbose. Print tracing information on the standard error as the program runs. Additional **–v** options increase the level of detail.

–V Version. Display the version of the program on the standard error and exit.

–1 Print fields one-per-line. Otherwise, fields are printed several to a line, to a maximum width of 60 characters.

–w *width*
Change the output to *width* characters.

FILES

/usr/share/lib/terminfo/?/* compiled terminal description database
/etc/termcap

SEE ALSO

curses(3V), **termcap**(5), **terminfo**(5V), **infocmp**(8V), **tic**(8V)

DIAGNOSTICS

tgetent failed with return code n
The termcap entry is not valid. In particular, check for an invalid '**tc=**' entry.

unknown type given for the termcap code *cc*.
The termcap description had an entry for *cc* whose type was not boolean, numeric or string.

wrong type given for the boolean (numeric, string) termcap code *cc*.
The boolean **termcap** entry *cc* was entered as a numeric or string capability.

the boolean (numeric, string) termcap code *cc* **is not a valid name.**
An unknown **termcap** code was specified.

tgetent failed on TERM=term.
The terminal type specified could not be found in the **termcap** file.

TERM=term: cap *cc* **(info** *ii***) is**
The **termcap** code was specified as a null string. The correct way to cancel an entry is with an '**@**', as in '**:bs@:**'. Giving a null string could cause incorrect assumptions to be made by the software which uses **termcap** or **terminfo**.

a function key for *cc* was specified, but it already has the value *vv*. When parsing the **ko** capability, the key *cc* was specified as having the same value as the capability *cc*, but the key *cc* already had a value assigned to it.

the unknown termcap name *cc* was specified in the ko termcap capability.
A key was specified in the **ko** capability which could not be handled.

the *vi* character v (info *ii*) has the value *xx*, but ma gives n.
The **ma** capability specified a function key with a value different from that specified in another setting of the same key.

the unknown *vi* key v was specified in the ma termcap capability.
A **vi**(1) key unknown to **captoinfo** was specified in the **ma** capability.

Warning: termcap sg (*nn*) and termcap ug (*nn*) had different values.
terminfo assumes that the **sg** (now **xmc**) and **ug** values were the same.

Warning: the string produced for *ii* may be inefficient.
The parameterized string being created should be rewritten by hand.

Null termname given.
The terminal type was null. This is given if the environment variable **TERM** is not set or is null.

cannot open *filename* for reading.
The specified file could not be opened.

WARNINGS

Certain **termcap** defaults are assumed to be true. The bell character (**terminfo bel**) is assumed to be ^G. The linefeed capability (**termcap nl**) is assumed to be the same for both **cursor_down** and **scroll_forward** (**terminfo cud1** and **ind**, respectively.) Padding information is assumed to belong at the end of the string.

The algorithm used to expand parameterized information for **termcap** fields such as **cursor_position** (**termcap cm**, **terminfo cup**) can sometimes produce a string that may not be optimal. In particular, the rarely used **termcap** operation *%n* produces strings that are especially long. Most occurrences of these non-optimal strings will be flagged with a warning message and may need to be recoded by hand.

The short two-letter name at the beginning of the list of names in a **termcap** entry, a hold-over from an earlier version of the system, has been removed.

NAME

catman – create the cat files for the manual

SYNOPSIS

/usr/etc/catman [**–nptw**] [**–M directory**] [**–T** *tmac.an*] [*sections*]

DESCRIPTION

catman creates the preformatted versions of the on-line manual from the **nroff**(1) input files. Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, **catman** recreates the **whatis** database.

If there is one parameter not starting with a '–', it is taken to be a list of manual sections to look in. For example

catman 123

only updates manual sections **1**, **2**, and **3**.

If an unformatted source file contains only a line of the form '**.so manx/yyy.x**', a symbolic link is made in the catx or fmtx directory to the appropriate preformatted manual page. This feature allows easy distribution of the preformatted manual pages among a group of associated machines with **rdist**(1), since it makes the directories of preformatted manual pages self-contained and independent of the unformatted entries.

OPTIONS

–n Do not (re)create the **whatis** database.

–p Print what would be done instead of doing it.

–t Create **troff**ed entries in the appropriate **fmt** subdirectories instead of **nroff**ing into the **cat** subdirectories.

–w Only create the **whatis** database that is used by **whatis**(1) and the **man**(1) **–f** and **–k** options. No manual reformatting is done.

–M Update manual pages located in the specified **directory** (**/usr/man** by default).

–T Use **tmac.an** in place of the standard manual page macros.

ENVIRONMENT

TROFF The name of the formatter to use when the **–t** flag is given. If not set, '**troff**' is used.

FILES

/usr/[share]/man	default manual directory location
/usr/[share]/man/man?/*.*	raw (nroff input) manual sections
/usr/[share]/man/cat?/*.*	preformatted **nroff**ed manual pages
/usr/[share]/man/fmt?/*.*	preformatted **troff**ed manual pages
/usr/[share]/man/whatis	**whatis** database location
/usr/lib/makewhatis	command script to make whatis database

SEE ALSO

apropos(1), **man**(1), **nroff**(1), **rdist**(1), **troff**(1), **whatis**(1)

NOTES

If the **–n** option is specified, the **/usr/man/whatis** database is not created and the **apropos**, **whatis**, '**man –f**', and '**man –k**' commands will fail.

DIAGNOSTICS

man?/xxx.? (.so'ed from man?/yyy.?): No such file or directory
The file outside the parentheses is missing, and is referred to by the file inside them.

target of .so in man?/xxx.? must be relative to /usr/man
catman only allows references to filenames that are relative to the directory **/usr/man**.

opendir:man?: No such file or directory
A harmless warning message indicating that one of the directories **catman** normally looks for is missing.

***.*: No such file or directory**
A harmless warning message indicating **catman** came across an empty directory.

NAME

change_login – control screen blanking and choice of login utility

SYNOPSIS

change_login

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

To prolong the life of your monitor, your Sun386i system turns off the screen display if you have not used the keyboard or mouse for 30 minutes or more. To see the screen again, simply move the mouse on the pad or press any key. This feature is normally enabled automatically when you log in, but you can control it using the **change_login** command as explained below.

This command also determines whether you log into your workstation using the Sun386i login screen, **logintool**(8) or through a traditional **login:** prompt.

The screen blanking choices available with **change_login** are:

1. Logintool and Sun Logo screenblank
Enables screen blanking. When blank, the system displays the Sun logo moving randomly around an otherwise dark screen.

2. Logintool and video-off screenblank
Shuts off the video output to your monitor when the screen goes blank. This is the most efficient type of screen blanking. The Desktop is almost instantly redisplayed when you move the mouse or begin typing.

3. Logintool and no screenblank
Retains the login screen, but disables screen blanking.

4. No Logintool and no screenblank
Disables both the login screen and screen blanking.

EXAMPLE

The following is an example of **change_login**. Notice you must be super-user to use this command.

```
example# change_login

This program will check what login and screenblank
options are set on this workstation, and allow you
to choose other options, if you are logged in as superuser.

Do you want to do this? [y or n]: y

This workstation is set up to use logintool and
a screenblank program that displays a Sun logo graphic.

These are the available options:

+ 1. Logintool and Sun Logo screenblank
  2. Logintool and video-off screenblank
  3. Logintool and no screenblank
  4. No Logintool and no screenblank

+ indicates the current configuration

You must be logged in as superuser to change the current setting.
```

Follow the instructions in *Sun386i System Setup and Maintenance* or *Sun386i Advanced Administration* to shut down and then restart your system. The setting chosen in the above example will not be enabled until you have restarted your system.

SEE ALSO

login(1), **screenblank**(1), **su**(1V), **logintool**(8)

Sun386i Advanced Skills
Sun386i System Setup and Maintenance
Sun386i Advanced Administration

NAME

chown – change owner

SYNOPSIS

/usr/etc/chown [**–fHR**] *owner*[*.group*] *filename* ...

DESCRIPTION

chown changes the owner of the *filenames* to *owner*. The owner may be either a decimal user ID (UID) or a login name found in the password file. An optional *group* may also be specified. The group may be either a decimal group ID (GID) or a group name found in the GID file.

Only the super-user can change owner, in order to simplify accounting procedures.

OPTIONS

–f Do not report errors.

–R Recursively descend into directories setting the ownership of all files in each directory encountered. When symbolic links are encountered, their ownership is changed, but they are not traversed.

FILES

/etc/passwd password file

SEE ALSO

chgrp(1), **chown**(2V), **group**(5), **passwd**(5)

NAME

chroot – change root directory for a command

SYNOPSIS

/usr/etc/chroot *newroot command*

DESCRIPTION

The given command is executed *relative* to the new root. The meaning of any initial '/' (slashes) in path names is changed for a command and any of its children to *newroot*. Furthermore, the initial working directory is *newroot*.

Input and output redirections on the command line are made with respect to the *original* root:

chroot newroot command >x

creates the file *x* relative to the original root, not the new one.

This command is restricted to the super-user.

The new root path name is always relative to the current root: even if a **chroot** is already in effect; the *newroot* argument is relative to the current root of the running process.

SEE ALSO

chdir(2V)

BUGS

One should exercise extreme caution when referring to special files in the new root file system.

NAME

chrtbl – generate character classification table

SYNOPSIS

/usr/etc/chrtbl [*filename*]

DESCRIPTION

chrtbl converts a source description of a character classification table into a form that can be used by the character classification functions and multibyte functions (see **ctype**(3V) and **mblen**(3)). The source description is found in *filename*. If *filename* is not given, or just given as '–', **chrtbl** reads its source description from the standard input.

chrtbl creates one or two output files, the second file is only created if the **model** token is specified. By default, these files are created in the current working directory. The first file, named by the **chrclass** token, is always produced and contains the character classification information for all single-byte (7-bit and 8-bit) character code-sets described by one setting of the **LC_CTYPE** category of locale. The second file, created if the **model** token is specified, contains information relating to details of width and structure of the coded character set currently under definition. The second file is named by appending '**.ci**'. to the value specified by the **chrclass** token.

The first output file contains a binary form of the character classification information described in *filename*. It is structured in such a way that it can be used at run-time to replace the active version of the **ctype[ ]** array in the C-library, For it to be understood at run-time, the output file must be moved to the **/usr/share/lib/locale/LC_TYPE** or **/etc/locale** directory (see FILES below) by the super-user or a member of group **bin**. This file must be readable by user, group, and other; no other permission should be set.

filename contains a sequence of tokens in any order after the **chrclass** token, each separated by one or more NEWLINE characters or comment lines. The tokens recognized by **chrtbl** are as follows:

chrclass *name*

name is the filename or pathname of the character classification file. This is a mandatory token. It must be the first token to be defined, and is usually given the name that relates to a valid setting of the **LC_CTYPE** category of locale.

model *name,args*

This optional token chooses the type of character code-set announcement mechanism associated with the character classification table generated by **chrtbl**. The name of the file created by this token is the name specified by the **chrclass** token, concatenated with a '**.ci**'. The arguments to **model** must be one of the following:

euc x,y,z

The model file contains information describing the required setting for the Extended Unix code-set announcement mechanism. *x,y,z* relate to the storage widths (in bytes) of EUC code-sets 1, 2 and 3 respectively.

xccs The model file contains information describing the Xerox Character Code Standard (XC1-3-3-0) announcement mechanism. There are no additional arguments required.

iso2022 g0,g1,g2,g3 x

The model file contains information describing a generative version of the ISO-2022 code set announcement mechanism. The multibyte functions driven by this model are capable of handling the standard one or more byte escape sequences as well as all of the standard shift functions. The four arguments *g0,g1,g2,g3* define the default width (in bytes) of the four designations (respectively) available under ISO-2022, Maximum integer value of any of these arguments is 2. The fianl argument *x* is mandatory and must be set to either 7 or 8. It selects the default bit-width of each byte on input and output to/from the multibyte functions.

If the **model** token is declared without arguments, then it is assumed that there is a set of user-defined rules for character code-set announcement. This is noted in the output file and will be later used to fold in user-defined code into the multibyte functions in the C-library (see **mblen**(3)).

isupper Character codes to be classified as upper-case letters.

islower Character codes to be classified as lower-case letters.

isdigit Character codes to be classified as numeric.

isspace Character codes to be classified as a spacing (delimiter) character.

ispunct Character codes to be classified as a punctuation character.

iscntrl Character codes to be classified as a control character.

isblank Character code for the space character.

isxdigit Character codes to be classified as hexadecimal digits.

ul Relationship between upper- and lower-case characters.

Any lines with the number sign (#) in the first column are treated as comments and are ignored. Blank lines are also ignored.

A character can be represented as a hexadecimal or octal constant (for example, the letter **a** can be represented as **0x61** in hexadecimal or **0141** in octal). Hexadecimal and octal constants may be separated by one or more space and tab characters.

The dash (–) may be used to indicate a range of consecutive numbers. Zero or more space characters may be used for separating the dash character from the numbers.

The backslash character (\) is used for line continuation. Only a RETURN is permitted after the backslash character.

The relationship between upper- and lower-case letters (**ul**) is expressed as ordered pairs of octal and hexadecimal constants:

<upper-case_character lower-case_character>

These two constants may be separated by one or more space characters. Zero or more space characters may be used for separating the angle brackets (< >) from the numbers.

EXAMPLES

The following is an example of an input file used to create the ASCII code set definition table on a file named **ascii**.

```
chrclass    ascii
isupper     0x41 – 0x5a
islower     0x61 – 0x7a
isdigit     0x30 – 0x39
isspace     0x20 0x9 – 0xd
ispunct     0x21 – 0x2f 0x3a – 0x40 \
            0x5b – 0x60 0x7b – 0x7e
iscntrl     0x0 – 0x1f 0x7f
isblank     0x20
isxdigit    0x30 – 0x39 0x61 – 0x66 \
            0x41 – 0x46
ul          <0x41 0x61> <0x42 0x62> <0x43 0x63> \
            <0x44 0x64> <0x45 0x65> <0x46 0x66> \
            <0x47 0x67> <0x48 0x68> <0x49 0x69> \
            <0x4a 0x6a> <0x4b 0x6b> <0x4c 0x6c> \
            <0x4d 0x6d> <0x4e 0x6e> <0x4f 0x6f> \
            <0x50 0x70> <0x51 0x71> <0x52 0x72> \
```

**<0x53 0x73> <0x54 0x74> <0x55 0x75> **
**<0x56 0x76> <0x57 0x77> <0x58 0x78> **
<0x59 0x79> <0x5a 0x7a>

FILES

/usr/share/lib/locale/LC_CTYPE/*	run-time location of the character classification tables generated by **chrtbl**
/etc/locale/LC_CTYPE/*	location for private versions of the classification tables generated by **chrtbl**

SEE ALSO

ctype(3V), **environ**(5V)

DIAGNOSTICS

The error messages produced by **chrtbl** are intended to be self-explanatory. They indicate input errors in the command line or syntactic errors encountered within the input file.

NAME

client – add or remove diskless Sun386i systems

SYNOPSIS

client [**–a** *arch*] [**–h** *hostid*] [**–o** *os*] [**–q**] [**–t** *minutes*] **add** *bootserver client etheraddress ipaddress*

client remove *client*

client modify *client* [**diskful** | **diskless** | **slave**]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

client can be used to manually add and remove diskless clients of a PNP boot server. After successful completion of the command, the diskless client can boot. Only users in the *networks* group (group 12) on the boot server are allowed to change configurations using this utility. **client** can be invoked from any system on the network.

The boot server of a system is the only machine truly required for that system to boot to the point of allowing user logins; it must accordingly provide name, booting, and time services. Diskless clients can provide none of these services themselves. Diskful clients, however can provide most of their own boot services. Network clients only need name and time services from the network, and can use any boot server.

To add a diskless client, use the **add** operation. To remove a diskless, diskful, or network client, use the **remove** operation. To change a system's network role, use the **modify** operation.

A server can reject a configuration request if it is disallowed by the contents of the **bootservers** map (e.g., too many clients would be configured, or too little free space would be left on the server), or if no system software for the client is available.

OPTIONS

–a *arch* Specifies the architecture code of the client; it defaults to **s386**. (Note: architecture codes are different from architecture names. Architecture codes are used in diskless booting, and are at most five characters in length, while architecture names can be longer.)

–h *hostid* Specifies the host ID of the client; if supplied, it is used as the root password for the system. It defaults to the null string.

–o *os* Specifies the operating system; defaults to 'unix'. This is currently used only to construct the system's *publickey* data, where applicable; this is never done if the system has no *hostid* specified.

–q Quiet. Displays only error messages.

–t *minutes* Sets the RPC timeout to the number of minutes indicated; this defaults to 15 minutes. If the bootserver takes more time than this to complete, **client** will exit. Unless the server has already completed setup, but not yet sent status to **client**, this will cause the bootserver to back out of the setup, deallocating all assigned resources.

SEE ALSO

publickey(5) **netconfig**(8C), **pnpd**(8C)

BUGS

Unless the *hostid* is assigned, the root filesystem for the diskless client is not set up beyond copying the **proto** and **boot** files into it. This means that **netconfig** will often handle other parts of the setup.

NAME

clri – clear inode

SYNOPSIS

/usr/etc/clri *filesystem i-number...*

DESCRIPTION

Note: **clri** has been superseded for normal file system repair work by **fsck**(8).

clri writes zeros on the inodes with the decimal *i-numbers* on the *filesystem*. After **clri,** any blocks in the affected file will show up as "missing" in an **icheck**(8) of the *filesystem.*

Read and write permission is required on the specified file system device. The inode becomes allocatable.

The primary purpose of this routine is to remove a file which for some reason appears in no directory. If it is used to zap an inode which does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated inode, so the whole cycle is likely to be repeated again and again.

SEE ALSO

icheck(8) **fsck**(8)

BUGS

If the file is open, **clri** is likely to be ineffective.

NAME

colldef – convert collation sequence source definition

SYNOPSIS

/usr/etc/colldef *filename*

DESCRIPTION

colldef converts a collation sequence source definition into a format usable by the **strxfrm**( ) and **strcoll**(3) functions. It is used to define the many ways in which strings can be ordered and collated.

colldef reads the collation sequence source definition from the standard input and stores the converted definition in *filename*.

The collation sequence definition specifies a set of collating elements and the rules defining how strings containing these should be ordered. This is most useful for different language definitions. The rules provide the following capabilities:

1-to-Many mapping
: A single character is mapped into a string of collating elements.

Many-to-1 mapping
: A string of two or more characters is mapped as a single collating element.

Null string mapping
: A character, or string of characters, is mapped to a null collating element (that is, will be ignored).

Equivalence class definition.
: A collection of characters that have the same value.

Secondary ordering within equivalence class.

USAGE

The following keywords may be used in the input file *filename*.

charmap
: Optional keyword. Defines where a mapping of the character and collating element symbols to the actual character encoding can be found.

substitute
: Optional keyword. Defines a one-to-many mapping between a single byte and a character string.

order Mandatory keyword. Defines the primary and secondary ordering of collating elements within this collation table.

EXIT STATUS

colldef exits with the following values:

0 No errors were found and the output was successfully created.

>0 Errors were found.

FILES

/etc/locale/LC_COLLATE/*locale/domain*
: standard private location for collation orders under the *locale* locale

/usr/share/lib/locale/LC_COLLATE
: standard shared location for collation orders under the *locale* locale

SEE ALSO

strcoll(3)
System Services Overview

NAME

comsat, in.comsat – biff server

SYNOPSIS

/usr/etc/in.comsat

DESCRIPTION

comsat is the server process which listens for reports of incoming mail and notifies users who have requested to be told when mail arrives. It is invoked as needed by **inetd**(8C), and times out if inactive for a few minutes.

comsat listens on a datagram port associated with the **biff**(1) service specification (see **services**(5)) for one line messages of the form

user@mailbox-offset

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a **'biff y'**), the *offset* is used as a seek offset into the appropriate mailbox file and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the **From**, **To**, **Date**, or **Subject** lines are not printed when displaying the message.

FILES

/etc/utmp	to find out who's logged on and on what terminals

SEE ALSO

biff(1), **services**(5), **inetd**(8C)

BUGS

The message header filtering is prone to error.

The notification should appear in a separate window so it does not mess up the screen.

NAME

config – build system configuration files

SYNOPSIS

/usr/etc/config [**–fgnp**] [**–o** *obj_dir*] *config_file*

DESCRIPTION

config does the preparation necessary for building a new system kernel with **make**(1). The *config_file* named on the command line describes the kernel to be made in terms of options you want in your system, size of tables, and device drivers to be included. When you run **config**, it uses several input files located in the current directory (typically the **conf** subdirectory of the system source including your *config_file*). The format of this file is described below.

If the directory named *../config_file* does not exist, **config** will create one. One of **config**'s output files is a makefile which you use with **make**(1) to build your system.

You use **config** as follows. Run **config** from the **conf** subdirectory of the system source (in a typical Sun environment, from **/usr/share/sys/sun[2 3 4]/conf**):

example# /usr/etc/config config_file
Doing a "make depend"
example# cd ../config_file
example# make
... lots of output...

While **config** is running watch for any errors. Never use a kernel which **config** has complained about; the results are unpredictable. If **config** completes successfully, you can change directory to the *../config_file* directory, where it has placed the new makefile, and use **make** to build a kernel. The output files placed in this directory include **ioconf.c**, which contains a description of I/O devices attached to the system; **mbglue.s**, which contains short assembly language routines used for vectored interrupts, a makefile, which is used by **make** to build the system; a set of header files (*device_name*.**h**) which contain the number of various devices that may be compiled into the system; and a set of swap configuration files which contain definitions for the disk areas to be used for the root file system, swapping, and system dumps.

Now you can install your new kernel and try it out.

OPTIONS

–f Set up the makefile for fast builds. This is done by building a **vmunix.o** file which includes all the **.o** files which have no source. This reduces the number of files which have to be **stat**ed during a system build. This is done by prelinking all the files for which no source exists into another file which is then linked in place of all these files when the kernel is made. This makefile is faster because it does not **stat** the object files during the build.

–g Get the current version of a missing source file from its SCCS history, if possible.

–n Do not do the '**make depend**'. Normally **config** will do the '**make depend**' automatically. If this option is used **config** will print '**Don't forget to do a "make depend"**' before completing as a reminder.

–p Configure the system for profiling (see **kgmon**(8) and **gprof**(1)).

–o *obj_dir*
Use *../obj_dir* instead of **../OBJ** as the directory to find the object files when the corresponding source file is not present in order to generate the files necessary to compile and link your kernel.

USAGE

Input Grammar

In the following descriptions, a number can be a decimal integer, a whole octal number or a whole hexadecimal number. Hex and octal numbers are specified to **config** in the same way they are specified to the C compiler, a number starting with **0x** is a hex number and a number starting with just a **0** is an octal number.

Comments are begin with a # character, and end at the next NEWLINE. Lines beginning with TAB characters are considered continuations of the previous line. Lines of the configuration file can be one of two basic types. First, there are lines which describe general things about your system:

machine "*type***"**

This is system is to run on the machine type specified. Only one machine type can appear in the config file. The legal *type*s for a Sun system are **sun2, sun3, sun4,** and **sun386.** Note: the double quotes around *type* are part of the syntax, and must be included.

cpu "*type***"**

This system is to run on the cpu type specified. More than one cpu type can appear in the config file. Legal *type*s for a **sun2** machine are noted in the annotated config file in *Installing SunOS 4.1*.

ident *name*

Give the system identifier — a name for the machine or machines that run this kernel. Note that *name* must be enclosed in double quotes if it contains both letters and digits. Also, note that if *name* is **GENERIC,** you need not include the '**options GENERIC**' clause in order to specify '**swap generic**'.

maxusers *number*

The maximum expected number of simultaneously active user on this system is *number*. This number is used to size several system data structures.

options *optlist*

Compile the listed options into the system. Options in this list are separated by commas. A line of the form:

options FUNNY,HAHA

yields

–DFUNNY –DHAHA

to the C compiler. An option may be given a value, by following its name with = (equal sign) then the value enclosed in (double) quotes. None of the standard options use such a value.

In addition, options can be used to bring in additional files if the option is listed in the **files** files. All options should be listed in upper case. In this case, no corresponding *option*.**h** will be created as it would be using the corresponding *pseudo-device* method.

config *sysname config_clauses...*

Generate a system with name *sysname* and configuration as specified in *config-clauses*. The *sysname* is used to name the resultant binary image and per-system swap configuration files. The *config_clauses* indicate the location for the root file system, one or more disk partitions for swapping and paging, and a disk partition to which system dumps should be made. All but the root device specification may be omitted; **config** will assign default values as described below.

root A root device specification is of the form '**root on** *xy0d*'. If a specific partition is omitted — for example, if only **root on xy0** is specified — the '**a**' partition is assumed. When a generic system is being built, no root specification should be given; the root device will be defined at boot time by prompting the console.

swap To specify a swap partition, use a clause of the form: '**swap on** *partition*'. Swapping areas may be almost any size. Partitions used for swapping are sized at boot time by the system; to override dynamic sizing of a swap area the number of sectors in the swap area can be specified in the config file. For example, '**swap on** *xy0b* **size 99999**' would configure a swap partition with 99999 sectors. If **swap generic** or no *partition* is specified with **on**, partition b on the root device is used. For dataless clients, use '**swap on type nfs**'.

To configure multiple swap partitions, specify multiple '**swap on**' clauses. For example:

config vmunix swap on xy0 swap on xy1

dumps The location to which system dumps are sent may be specified with a clause of the form '**dumps on** *xyl*'. If no dump device is specified, the first swap partition specified is used. If a device is specified without a particular partition, the '**b**' partition is assumed. If a generic configuration is to be built, no dump device should be specified; the dump device will be assigned to the swap device dynamically configured at boot time. Dumps are placed at the end of the partition specified. Their size and location is recorded in global kernel variables *dumpsize* and *dumplo*, respectively, for use by **savecore**(8).

Device names specified in configuration clauses are mapped to block device major numbers with the file **devices.***machine*, where *machine* is the machine type previously specified in the configuration file. If a device name to block device major number mapping must be overridden, a device specification may be given in the form '**major** *x* **minor** *y*'.

The second group of lines in the configuration file describe which devices your system has and what they are connected to (for example, a Xylogics 450 Disk Controller at address 0xee40 in the Multibus I/O space). These lines have the following format:

dev_type dev_name **at** *con_dev more_info*

dev_type is either **controller, disk, tape, device,** or **pseudo-device.** These types have the following meanings:

controller A disk or tape controller.

disk or **tape** Devices connected to a controller.

device Something "attached" to the main system bus, like a cartridge tape interface.

pseudo-device A software subsystem or driver treated like a device driver, but without any associated hardware. Current examples are the pseudo-tty driver and various network subsystems. For pseudo-devices, **more_info** may be specified as an integer, that gives the value of the symbol defined in the header file created for that device, and is generally used to indicate the number of instances of the pseudo-device to create.

dev_name is the standard device name and unit number (if the device is not a **pseudo-device**) of the device you are specifying. For example, **xyc0** is the *dev_name* for the first Xylogics controller in a system; **ar0** names the first quarter-inch tape controller.

con_dev is what the device you are specifying is connected to. It is either nexus?, a bus type, or a controller. There are several bus types which are used by **config** and the kernel.

The different possible bus types are:

obmem	On board memory
obio	On board io
mbmem	Multibus memory (**sun2** system only)
mbio	Multibus io (**sun2** system only)
vme16d16 (vme16)	16 bit VMEbus/ 16 bit data
vme24d16 (vme24)	24 bit VMEbus/ 16 bit data
vme32d16	32 bit VMEbus/ 16 bit data (**sun3** system only)
vme16d32	16 bit VMEbus/ 32 bit data (**sun3** system only)
vme24d32	24 bit VMEbus/ 32 bit data (**sun3** system only)
vme32d32 (vme32)	32 bit VMEbus/ 32 bit data (**sun3** system only)

All of these bus types are declared to be connected to nexus. The devices are hung off these buses. If the bus is wildcarded, then the autoconfiguration code will determine if it is appropriate to probe for the device on the machine that it is running on. If the bus is numbered, then the autoconfiguration code will only look for that device on machine type **N**. In general, the Multibus and VMEbus bus types are always wildcarded.

more_info is a sequence of the following:

csr *address* — Specify the address of the **csr** (command and status registers) for a device. The **csr** addresses specified for the device are the addresses within the bus type specified.

The **csr** address must be specified for all controllers, and for all devices connected to a main system bus.

drive *number* — For a disk or tape, specify which drive this is.

flags *number* — These flags are made available to the device driver, and are usually read at system initialization time.

priority *level* — For devices which interrupt, specify the interrupt level at which the device operates.

vector *intr number* [*intr number* . . .]
For devices which use vectored interrupts on VMEbus systems, *intr* specify the vectored interrupt routine and *number* the corresponding vector to be used (0x40-0xFF).

A **?** may be substituted for a number in two places and the system will figure out what to fill in for the **?** when it boots. You can put question marks on a *con_dev* (for example, at virtual '?'), or on a drive number (for example, drive '?'). This allows redundancy, as a single system can be built which will boot on different hardware configurations.

The easiest way to understand **config** files it to look at a working one and modify it to suit your system. Good examples are provided in *Installing SunOS 4.1*.

FILES

Files in **/usr/share/sys/sun[2 3 4]/conf** which may be useful for developing the *config_file* used by **config** are:

GENERIC — These are generic configuration files for either a Sun-2 or Sun-3 system. They contain all possible device descriptions lines for the particular architecture.

README — File describing how to make a new kernel.

As shipped from Sun, the files used by **/usr/etc/config** as input are in the **/usr/include/sys/conf** directory:

config_file	System-specific configuration file
Makefile.src	Generic prototype makefile for Sun–[23] systems
files	List of common files required to build a basic kernel
devices	Name to major device mapping file for Sun–[23] systems

/usr/etc/config places its output files in the *../config_file* directory:

mbglue.s	Short assembly language routines used for vectored interrupts
ioconf.c	Describes I/O devices attached to the system
makefile	Used with **make**(1) to build the system
device_name.**h**	a set of header files (various *device_name*'s) containing devices which can be compiled into the system

SEE ALSO

gprof(1), **make**(1), **kgmon**(8), **savecore**(8)

The SYNOPSIS portion of each device entry in Section 4 of this manual.

Installing SunOS 4.1
System and Network Administration

NAME

copy_home – fetch default startup files for new home directories

SYNOPSIS

/home/*groupname***/copy_home /home/***groupname* **/home/***username*

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

Whenever **snap**(1) is used to add a new user account, the **copy_home** script in the selected primary group's home directory is executed to copy the default files to the new user's home directory, and also perform any additional custom setup.

copy_home copies default environment files, such as **.cshrc**, **.login**, and **.orgrc**, from a group's **defaults** directory to a new user's home directory. It is started by **user_agentd**(8) when **snap**(1) is used to create new home directories on a Sun386i home directory server.

Every new group created by **snap**(1) has a home directory, which can be accessed using **/home/***groupname*. **user_agentd**(8) copies the contents of the Sun386i's default group, **/home/users**, into the home directory of the new group. This includes the **Welcome.txt** file, the **copy_home** script, and the **defaults** directory . **copy_home** can be modified to customize the default setup environment for new users in the group.

SEE ALSO

snap(1), **user_agentd**(8)

Sun386i SNAP Administration
Sun386i Advanced Administration

NAME

crash – examine system images

SYNOPSIS

/etc/crash [**–d** *dump-file*] [**–n** *namelist-file*] [**–w** *output-file*]

DESCRIPTION

crash examines the memory image of a live or a crashed system kernel. It displays the values of system control structures, tables, and other pertinent information.

OPTIONS

–d *dump-file* Specify the file containing the system memory image. The default is **/dev/mem**.

–n *namelist-file*

Specify the text file containing the symbol table for symbolic access to the memory image. The default is **/vmunix**. If a system image from another machine is to be examined, the image file must be copied from that machine.

–w *output-file* Specify a file for **crash** output. The default is the standard output.

USAGE

For commands that pertain to a process, the default process is the one currently running on a live system, or the one that was running at the time the system crashed.

If the contents of a table are being dumped, the default is all active table entries.

Numeric Notation

Depending on the command, numeric arguments are assumed to be in a specific base. Counts are assumed to be decimal. Addresses are always hexadecimal. Table addresses larger than the size of the specified table are interpreted as hexadecimal addresses; smaller arguments are assumed to be in decimal. The default base of any argument may be overridden; the C conventions for designating the base of a number are recognized. (A number that is usually interpreted as decimal will be interpreted as hexadecimal if it is preceded by **0x** and as octal if it is preceded by **0**. Decimal override is designated by **0d**, and binary by **0b**.)

Expressions

Many commands accept several forms of an argument. Requests for table information accept a table entry number, a physical address, a virtual address, a symbol, a range, or an expression. A range of slot numbers may be specified in the form *a–b* where *a* and *b* are decimal numbers. An expression consists of two operands and an operator. An operand may be an address, a symbol, or a number. The operator may be "+" (plus sign), "–" (minus sign), "*" (multiplication symbol), "/" (division symbol), "&" (logical AND), or "|" (logical OR). An operand which is a number should be preceded by a radix prefix if it is not a decimal number (**0** for octal, **0x** for hexidecimal, **0b** for binary). The expression must be enclosed in '()' (parentheses). Other commands accept any of these argument forms that are meaningful.

Two abbreviated arguments to **crash** commands are used throughout. Both accept data entered in several forms. A *table_entry* argument may be an address, symbol, range or expression that resolves to one of these. A *start_addr* argument may be an address, symbol, or expression that resolves to one of those.

Commands

? [**–w** *filename*]

List available commands.

–w *filename*

Redirect the output of a command to the named file. Corresponds to the **redirect** command.

!*command*

Escape to the shell to execute a command.

adv [**–ep**] [**–w** *filename*] [*table_entry*] ...
Print the advertise table.

–e Display every entry in a table.

–p Interpret all address arguments in the command line as physical addresses. With this option, all address and symbol arguments explicitly entered on the command line are interpreted as physical addresses. Corresponds to the **mode** command.

as [**–w**filename] [**–p**] *proc_entry* | *#pid* [**s**]]
Print the address space table.

base [**–w** *filename*] *number* ...
Print *number* in binary, octal, decimal, and hexadecimal. A number in a radix other then decimal should be preceded by a prefix that indicates its radix as follows: **0x**, hexidecimal; **0**, octal; and **0b**, binary.

buffer [**–w** *filename*] [*–format*] *bufferslot*
buffer [**–p**] [**–w** *filename*] [*–format*] *start_addr*
Alias: **b**.
Print the contents of a buffer in the designated format. The following format designations are recognized: **–b**, byte; **–c**, character; **–d**, decimal; **–x**, hexadecimal; **–o**, octal; **–r**, directory; and **–i**, inode. If no format is given, the previous format is used. The default format at the beginning of a **crash** session is hexadecimal.

bufhdr [**–fp**] [**–w** *filename*] [*table_entry*] ...
Alias: **buf**.
Print system buffer headers.

–f Display the full structure.

callout [**–w** *filename*]
Alias: **c**.
Print the callout table.

ctx [**–w***filename*] [[**–p**] *tbl_entry* ...]
Print the context table.

dbfree [**–w** *filename*]
Print free streams data block headers. If a class is entered, only data block headers for the class specified will be printed.

dblock [**–ep**] [**–w** *filename*] [*dblk_addr*] ...
Print allocated streams data block headers. If the class option (**–c**) is used, only data block headers for the class specified will be printed.

defproc [**–c**] [**–w** *filename*]
defproc [**–w** *filename*] [*slot*]
Set the value of the process slot argument. The process slot argument may be set to the current slot number (**–c**) or the slot number may be specified. If no argument is entered, the value of the previously set slot number is printed. At the start of a **crash** session, the process slot is set to the current process.

ds [**–w** *filename*] *virtual_address* ...
Print the data symbol whose address is closest to, but not greater than, the address entered.

file [**–ep**] [**–w** *filename*] [*table_entry*] ...
Alias: **f**.
Print the file table.

findaddr [**–w** *filename*] *table slot*
Print the address of *slot* in *table*. Only tables available to the **size** command are available to **findaddr**.

gdp [**–efp**] [**–w** *filename*] [*table_entry*] ...
Print the gift descriptor protocol table.

help [**–w** *filename*] *command* ...
Print a description of the named command, including syntax and aliases.

inode [**–f**] [**–w** *filename*] [*table_entry*] ...
Alias: **i**.
Print the inode table, including file system switch information.

kfp [**–r**] [**–s** *process*] [**–w** *filename*]
kfp [**–s** *process*] [**–w** *filename*] [*value*]
Print the frame pointer for the start of a kernel stack trace. The **kfp** value can be set using the value argument or the reset option (**–r**), which sets the **kfp** through the nvram. If no argument is entered, the current value of the **kfp** is printed.

–s *process* Specify a process slot other than the default. Corresponds to the **defproc** command.

linkblk [**–ep**] [**–w** *filename*] [*table_entry*] ...
Print the **linkblk** table.

map [**–w** *filename*] *mapname* ...
Alias: **m**.
Print the map structure of *mapname*.

mbfree [**–w** *filename*]
Print free streams message block headers.

mblock [**–ep**] [**–w** *filename*] [*mblk_addr*] ...
Print allocated streams message block headers.

mode [**–w** *filename*] [*mode*]
Set address translation of arguments to virtual (**v**) or physical (**p**) mode. If no mode argument is given, the current mode is printed. At the start of a **crash** session, the mode is virtual.

mount [**–p**] [**–w** *filename*] [*table_entry*] ...
Alias: **m**.
Print the mount table.

nm [**–w** *filename*] *symbol* ...
Print value and type for the given symbol.

od [**–p**] [**–w** *filename*] [*–format*] [*–mode*] [**–s** *process*] *start_addr* [*count*]
Alias: **rd**.
Print *count* values starting at the start address in one of the following formats:

–c character
–d decimal
–x hexadecimal
–o octal
–a ASCII
–h hexadecimal character

and one of the following modes:

–l long
–t short
–b byte

The default mode for character and ASCII formats is byte; the default mode for decimal, hexadecimal, and octal formats is long. The format **–h** prints both hexadecimal and character representations of the addresses dumped; no mode needs to be specified. When format or mode is omitted, the previous value is used. At the start of a **crash** session, the format is hexadecimal and the mode is long. If no count is entered, 1 is assumed.

page [**–e**] [**–w***filename*] [[**–p**] *tbl_entry*] ...
Alias: **p.**
Print the page structures.

pcb [**–w** *filename*] [*process*]
Print the process control block. If no arguments are given, the active **pcb** for the current process is printed. **–ep**

pment [**–p**] [**–w***filename*] *tbl_entry* ...
Print the page map entry table (not available on machines with a **sun3x** kernel architecture).

pmgrp [**–w***filename*] [[**–p**] *tbl_entry* ...]
Print the page map group table (not available on machines with a **sun3x** kernel architecture).

proc [**–fp**] [**–w** *filename*] [*#pid*] ... [*table_entry*] ...
proc [**–fr**] [**–w** *filename*]
Print the process table. Process table information may be specified in two ways. First, any mixture of table entries and process IDs (PID) may be entered. Each PID must be preceded by a '#' (pound sign). Alternatively, process table information for runnable processes may be specified with the runnable option (**–r**).

qrun [**–w** *filename*]
Print the list of scheduled streams queues.

queue [**–p**] [**–w** *filename*] [*queue_addr*] ...
Print stream queues.

quit Alias: **q.**
Terminate the **crash** session.

rcvd [**–efp**] [**–w** *filename*] [*table_entry*] ...
Print the receive descriptor table.

redirect [**–c**] [**–w** *filename*]
redirect [**–w** *filename*] [*filename*]
Alias: **rd.**
Used with a name, redirects output of a **crash** session to the named file. If no argument is given, the file name to which output is being redirected is printed. Alternatively, the close option (**–c**) closes the previously set file and redirects output to the standard output. To pipe output from a single **crash** command, use an exclamation point followed by a shell command:
crash-command **!** *shell-command*

This is not available when **–w** is in effect.

search [**–p**] [**–m** *mask*] [**–s** *process*] [**–w** *filename*] *pattern start_addr length*
Alias: **s.**
Print the words in memory that match *pattern*, beginning at the start address for *length* words. The mask is ANDed (&) with each memory word and the result compared against the pattern. The mask defaults to **0xffffffff**.

seg [**–w***filename*] [[**–p**] *proc_entry*]
seg [**–w***filename*] [*#procid* ...]
Print the segment table of process.

segdata [**–w***filename*] [[**–p**] *proc_entry*]
segdata [**–w***filename*] [*#procid* ...]
Print the segment data of process.

size [**–x**] [**–w** *filename*] [*structure_name* ...]
Print the size of the designated structure. The **–x** option prints the size in hexadecimal. If no argument is given, a list of the structure names for which sizes are available is printed.

sndd [**–efp**] [**–w** *filename*] [*table_entry*] ...
Print the send descriptor table.

srmount [**–ep**] [**–w** *filename*] [*table_entry*] ...
Print the server mount table.

stack [**–u**] [**–w** *filename*] [*process*]
stack [**–k**] [**–w** *filename*] [*process*]
stack [**–p**] [**–w** *filename*] **–i** *start_addr*]
Alias: **s.**
Dump stack. The **–u** option prints the user stack. The **–k** option prints the kernel stack. The **–i** option prints the interrupt stack starting at the start address. If no arguments are entered, the kernel stack for the current process is printed. The interrupt stack and the stack for the current process are not available on a running system.

status [**–w** *filename*]
Print system statistics.

stream [**–efp**] [**–w** *filename*] [*table_entry*] ...
Print the streams table.

strstat [**–w** *filename*]
Print streams statistics.

trace [**–r**] [**–w** *filename*] [*process*]
trace [**–p**] [**–w** *filename*] **–i** *start_addr*]
Alias: **t.**
Print stack trace. The **kfp** value is used with the **–r** option. The interrupt option prints a trace of the interrupt stack beginning at the start address. The interrupt stack trace and the stack trace for the current process are not available on a running system.

ts [**–w** *filename*] *virtual_address* ...
Print closest text symbol to the designated address.

user [**–f**] [**–w** *filename*] [*process*]
Alias: **u.**
Print the ublock for the designated process.

vfs [**–w***filename*] [[**–p**] *tbl_entry* ...]
Print the **vfs** table.

vnode [**–w***filename*] [[**–p**] *addr*]
Alias: **v.**
Print the vnode table.

vtop [**–s** *process*] [**–w** *filename*] *start_addr* ...
Print the physical address translation of the virtual start address.

FILES

/dev/mem	system image of currently running system
/var/crash/*machine***/vmcore.***N*	
/var/crash/*machine***/vmunix.***N*	

SEE ALSO

savecore(8)

NAME

cron – clock daemon

SYNOPSIS

/usr/etc/cron

DESCRIPTION

cron executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in **crontab** files in the directory **/var/spool/cron/crontabs**. Users can submit their own **crontab** files using the **crontab**(1) command. Commands that are to be executed only once may be submitted using the **at**(1) command.

cron only examines **crontab** files and **at** command files during process initialization and when a file changes using **crontab** or **at**. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

Since **cron** never exits, it should only be executed once. This is normally done by running **cron** from the initialization process through the file **/etc/rc**; see **init**(8). **/var/spool/cron/FIFO** is a FIFO file that **crontab** and **at** use to communicate with **cron**; it is also used as a lock file to prevent the execution of more than one **cron**.

FILES

/var/spool/cron	main cron directory
/var/spool/cron/FIFO	FIFO for sending messages to **cron**
/var/spool/cron/crontabs	directory containing **crontab** files

SEE ALSO

at(1), **crontab**(1), **sh**(1), **queuedefs**(5), **init**(8), **syslogd**(8)

DIAGNOSTICS

cron logs various errors to the system log daemon, **syslogd**(8), with a facility code of **cron**. The messages are listed here, grouped by severity level.

Err Severity

Can't create /var/spool/cron/FIFO: *reason*
: **cron** was unable to start up because it could not create **/var/spool/cron/FIFO**.

Can't access /var/spool/cron/FIFO: *reason*
: **cron** was unable to start up because it could not access **/var/spool/cron/FIFO**.

Can't open /var/spool/cron/FIFO: *reason*
: **cron** was unable to start up because it could not open **/var/spool/cron/FIFO**.

Can't start cron - another cron may be running (/var/spool/cron/FIFO exists)
: **cron** found that **/var/spool/cron/FIFO** already existed when it was started; this normally means that **cron** had already been started, but it may mean that an earlier **cron** terminated abnormally without removing **/var/spool/cron/FIFO**.

Can't stat /var/spool/cron/FIFO: *reason*
: **cron** could not get the status of **/var/spool/cron/FIFO**.

Can't change directory to *directory*:*reason*
: **cron** could not change to *directory*.

Can't read *directory*:*reason*
: **cron** could not read *directory*.

error reading message: *reason*
: An error occurred when **cron** tried to read a control message from **/var/spool/cron/FIFO**.

message received — bad format
A message was successfully read by **cron** from **/var/spool/cron/FIFO**, but the message was not of a form recognized by **cron.**

SIGTERM
received **cron** was told to terminate by having a SIGTERM signal sent to it.

cron could not unlink /var/spool/cron/FIFO: *reason*
cron was told to terminate, but it was unable to unlink **/var/spool/cron/FIFO** before it terminated.

********* CRON ABORTED **********
cron terminated, either due to an error or because it was told to.

Can't open queuedefs file *file*:*reason*
cron could not open a *queuedefs* file.

I/O error reading queuedefs file *file*:*reason*
An I/O error occurred while **cron** was reading a *queuedefs* file.

Using default queue definitions
An error occurred while trying to read a *queuedefs* file; the default queue definitions will be used.

Can't allocate *number* **bytes of space**
An internal error occurred in **cron** while trying to allocate memory.

Info Severity

queue **queue max run limit reached**
There were more jobs running or to be run in the queue *queue* than the maximum number specified. **cron** will wait until one of the currently-running jobs completes before starting to run a new one.

MAXRUN (25) procs reached
There were more than 25 jobs running or to be run by **cron**. **cron** will wait until one of the currently-running jobs completes before starting to run a new one.

***** cron started *****
cron started running.

> CMD: *pid queue command job*
A **cron** job was started, in queue *queue*, with process ID *pid*. *command* is the command to be run. For **at** or **batch** jobs, *job* is the job number.

> *user pid queue time job*
A **cron** job was started for user *user*, in queue *queue*, with process ID *pid*, at the date and time *time*. For **at** or **batch** jobs, *job* is the job number.

< *user pid queue time job status*
A **cron** job completed for user *user*, in queue *queue*, with process ID *pid*, at the date and time *time*. For **at** or **batch** jobs, *job* is the job number. If the command terminated with a non-zero exit status or a signal, *status* indicates the exit status or signal.

Notice Severity

Can't fork
An attempt to **fork (2)** to run a new job failed; **cron** will attempt again after a 30-second delay.

Warning Severity

Can't stat queuedefs file *file*:*reason*
cron could not get the status of a *queuedefs* file in order to determine whether it has changed. **cron** will assume it has changed and will reread it.

NAME

dbconfig – initializes the dial box

SYOPNSIS

/usr/etc/dbconfig *serial-device*

DESCRIPTION

dbconfig opens the designated serial port and sets its baud, parity and transmission rates. It also removes all STREAMS modules already pushed upon it (such as **ttcompat**(4M) and **ldterm**(4M)) and pushes the dial box STREAMS module "db" onto the device. **db** then holds the stream open to maintain this configuration.

If the device **/dev/dialbox** has not been created and linked to the serial port, **dbconfig** will fail.

FILES

/dev/dialbox

SEE ALSO

db(4M), **ldterm**(4M), **ttcompat**(4M), **dialtest**(6)

NAME

dcheck – file system directory consistency check

SYNOPSIS

/usr/etc/dcheck [**–i** *numbers*] [*filesystem*]

DESCRIPTION

Note: **dcheck** has been superseded for normal consistency checking by **fsck**(8).

dcheck reads the directories in a file system and compares the link-count in each inode with the number of directory entries by which it is referenced. If the file system is not specified, **dcheck** checks a set of default file systems.

dcheck is fastest if the raw version of the special file is used, since the i-list is read in large chunks.

OPTIONS

–i *numbers*

numbers is a list of i-numbers; when one of those i-numbers turns up in a directory, the number, the i-number of the directory, and the name of the entry are reported.

FILES

Default file systems vary with installation.

SEE ALSO

fs(5), **fsck**(8), **clri**(8), **icheck**(8), **ncheck**(8)

DIAGNOSTICS

When a file turns up for which the link-count and the number of directory entries disagree, the relevant facts are reported. Allocated files which have 0 link-count and no entries are also listed. The only dangerous situation occurs when there are more entries than links; if entries are removed, so the link-count drops to 0, the remaining entries point to thin air. They should be removed. When there are more links than entries, or there is an allocated file with neither links nor entries, some disk space may be lost but the situation will not degenerate.

BUGS

Since **dcheck** is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

Inode numbers less than 2 are invalid.

NAME

devinfo – print out system device information

SYNOPSIS

/usr/etc/devinfo [**–v**]

AVAILABILITY

This program is available on SPARCstation 1 systems only.

DESCRIPTION

devinfo displays the devices that the system knows about. The output will state the name of the device, its unit number, and whether a system device driver has claimed it. Since the internal system representation of this information is an *n*–ary tree, indentation is used to denote a parent-child relationship, and devices reported at the same indentation level are considered sibling devices.

OPTIONS

–v Report hardware specifications such as register addresses and interrupt priorities for each device.

EXAMPLE

The following example displays the format of **devinfo** output:

```
example% devinfo
Node 'Sun 4/60', unit #0 (no driver)
        Node 'options', unit #0 (no driver)
        Node 'zs', unit #0
        Node 'zs', unit #1
        Node 'fd', unit #0
        Node 'audio', unit #0
        Node 'sbus', unit #0
                Node 'dma', unit #0
                Node 'esp', unit #0
                        Node 'st', unit #1 (no driver)
                        Node 'st', unit #0
                        Node 'sd', unit #3
                        Node 'sd', unit #2
                        Node 'sd', unit #1
                        Node 'sd', unit #0
                Node 'le', unit #0
                Node 'bwtwo', unit #0
        Node 'auxiliary-io', unit #0
        Node 'interrupt-enable', unit #0
        Node 'memory-error', unit #0
        Node 'counter-timer', unit #0
        Node 'eeprom', unit #0
```

FILES

/dev/kmem to get kernel device information

NAME

devnm – device name

SYNOPSIS

/usr/etc/devnm [*name*]...

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

devnm identifies the special file associated with the mounted file system where each *name* argument resides. This command can be used to construct a mount table entry for the **root** file system.

EXAMPLE

If **/usr** is mounted on **/dev/dsk/c1d0s2**, then the command:

/usr/etc/devnm /usr

produces:

/dev/dsk/c1d0s2 usr

FILES

/dev/dsk/*
/etc/mtab

SEE ALSO

fstab(5) **mount**(8)

NAME

diskusg – generate disk accounting data by user

SYNOPSIS

diskusg [**–sv**] [**–p** *filename*] [**–u** *filename*] [*filename* ...]

DESCRIPTION

diskusg generates intermediate disk accounting information from data in *filename*, or the standard input if *filename* is omitted. **diskusg** displays one line per user on the standard output in the following format:

uid login #blocks

uid is the numerical user ID of the user. *login* is the user's login name. *#blocks* is the total number of disk blocks allocated to the user.

diskusg normally reads only the i-nodes of file systems for disk accounting. In this case, *filename*s are the special filenames of these devices.

The output of **diskusg** is normally the input to **acctdisk** (see **acct**(8)) which generates total accounting records that can be merged with other accounting records. **diskusg** is normally run in **dodisk** (see **acctsh**(8)).

OPTIONS

–s	The input data is already in **diskusg** output format; combine all lines for a single user into a single line.
–v	Print a list to the standard error of all files that are not charged to any user.
–p *filename*	Use *filename* as the name of the password file to generate login names. **/etc/passwd** is used by default.
–u *filename*	Write records to *filename* of files that are not charged to any user. Records consist of the special file name, the i-node number, and the user ID.

EXAMPLES

The following example generates daily disk accounting information:

```
for i in /dev/xy0a /dev/xy0g /dev/xy1g; do
        diskusg $i > dtmp.‘basename $i‘ &
done
wait
diskusg –s dtmp.* | sort +0n +1 | acctdisk > disktacct
```

FILES

/etc/passwd	used for user ID to login name conversions

SEE ALSO

acct(5), **acct**(8), **acctsh**(8)

NAME

dkctl – control special disk operations

SYNOPSIS

/usr/etc/dkctl *disk command*

DESCRIPTION

dkctl is used to enable or disable special disk operations. In particular the enabling or disabling of verified writes (write check functionality) is controlled by this program.

The *disk* specification here is a disk name of the form */dev/rxxnp*, where *xx* is the controller device abbreviation (xy, sd, etc.), *n* is the disk number, and *p* is the partition to which the operation applies. The *partition* specification is simply the letter used to identify that partition in the standard UNIX system nomenclature.

SUPPORTED COMMANDS

wchk This function enables write checking for disks that support it for the named disk partition. This means that for partitions of disks with this feature enabled, all writes are *verified* to have been correctly written on the disk. This operation emphasizes data reliability over performance, although for each implementation, the fastest reasonable method will be used (i.e., implemented in hardware, if possible).

–wchk This disables write check functionality for the named disk partition.

BUGS

Use of the **dkctl** command requires super-user permissions.

There are many other features this program could control, and may in the future.

FILES

/dev/r*xxnp*

SEE ALSO

dkio(4S), **sd**(4S), **xy**(4S)

NAME

dkinfo – report information about a disk's geometry and partitioning

SYNOPSIS

/usr/etc/dkinfo *disk* [*partition*]

DESCRIPTION

dkinfo gives the total number of cylinders, heads, and sectors or tracks on the specified *disk*, and gives this information along with the starting cylinder for the specified *partition*. If no *partition* is specified on the command line, **dkinfo** reports on all partitions.

The *disk* specification here is a disk name of the form *xxn*, where *xx* is the controller device abbreviation (ip, xy, etc.) and *n* is the disk number. The *partition* specification is simply the letter used to identify that partition in the standard UNIX system nomenclature. For example, '**/usr/etc/dkinfo xy0**' reports on the first disk in a system controlled by a Xylogics controller; '**/usr/etc/dkinfo xy0g**' reports on the seventh partition of such a disk.

EXAMPLE

A request for information on my local disk, an 84 MByte disk controlled by a Xylogics 450 controller, might look like this:

```
#/usr/etc/dkinfo xy0
xy0: Xylogics 450 controller at addr ee40, unit # 0
586  cylinders 7 heads 32 sectors/track
a: 15884 sectors (70 cyls, 6 tracks, 12 sectors)
starting cylinder 0
b: 33440 sectors (149 cyls, 2 tracks)
starting cylinder 71
c: 131264 sectors (586 cyls)
starting cylinder 0
d: No such device or address
e: No such device or address
f: No such device or address
g: 81760 sectors (365 cyls)
starting cylinder 221
h: No such device or address
#
```

FILES

/dev/r*xxnp*

SEE ALSO

dkio(4S), **format**(8S)

NAME

dmesg – collect system diagnostic messages to form error log

SYNOPSIS

/usr/etc/dmesg [–]

DESCRIPTION

Note: **dmesg** is obsoleted by **syslogd**(8) for maintenance of the system error log.

dmesg looks in a system buffer for recently printed diagnostic messages and prints them on the standard output. The messages are those printed or logged by the system when errors occur. If the '–' flag is given, then **dmesg** computes (incrementally) the new messages since the last time it was run and places these on the standard output.

FILES

/var/adm/msgbuf scratch file for memory of '–' option

SEE ALSO

syslogd(8)

NAME

dname – print RFS domain and network names

SYNOPSIS

dname [**–adn**] [**–D** *domain*] [**–N** *netspec*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

dname prints or defines a host's Remote File Sharing (RFS) domain name or the network used by RFS as transport provider.

When **dname** is used to change a domain name, the host's password is removed. The administrator will be prompted for a new password the next time RFS is started. See **rfstart**(8).

If **dname** is used with no options, it defaults to '**dname –d**'.

You cannot use the **–D** or **–N** options while RFS is running.

OPTIONS

–a Print both the domain name and network name.

–d Print the domain name.

–n Print the network name.

–D *domain*
Set the domain name for the host. *domain* must consist of no more than 14 characters, consisting of any combination of letters (upper and lower case), digits, hyphens (–), and underscores (_). This option is restricted to the super-user.

–N *netspec*
Set the network specification used for RFS. *netspec* is the network device name, relative to the **/dev** directory. For example, the TCP transport device, **/dev/tcp** uses **tcp**. This option is restricted to the super-user.

SEE ALSO

rfstart(8)

NOTES

This domain name is not related to the Network Interface Service (NIS) domain name. Note: NIS was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

dorfs – initialize, start and stop RFS automatically

SYNOPSIS

dorfs init *domain netspec* [*address*]
dorfs start [**–v**]
dorfs stop

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

dorfs sets up necessary environment to run Remote File Sharing (RFS). You can also use it to start or stop RFS automatically, after its environment is initialized. The environment only needs to be set up once and **/usr/nserve/rfmaster** must exist before the environment is initialized. Descriptions of **/usr/nserve/rfmaster** are in **rfmaster**(5). You must be the super-user to run this command.

USAGE

Subcommands

init *domain netspec* [*address*]

domain is the name of the RFS domain. *netspec* is the name of a device file in the **/dev** directory which represents the streams-based transport provider on which RFS will run. Currently, **tcp** is the only accepted value for this field. *address* is the optional **tcp** port number on which the listener will listen. If unspecified, it defaults to **0x1450**. This subcommand only needs to be run once to initialize the environment. You do not need to rerun **dorfs** with the **init** argument, unless you want to change *netspec*. **/usr/nserve/rfmaster** must exists before you run this command to initialize the environment. To reinitialize the environment, you need to remove **/usr/nserve/***domain*, **/usr/nserve/***netspec*, **/var/net/nls/***netspec/address* and **/var/net/nls/***netspec***/dbf** beforehand.

start [**–v**]

Start RFS automatically. It also automatically advertises resources that are stored in **/etc/rstab** and mounts RFS resources that are stored in **/etc/fstab**.

–v Verify clients on mounts (see '**rfstart –v**').

stop

Takes down RFS by forced unmounting of all advertised resources, umounting all remotely mounted resources, executing **rfstop**, and stopping **listener**.

FILES

/etc/advtab
/etc/rstab
/var/net/nls/tcp/addr
/var/net/nls/tcp/dbf
/usr/nserve/domain
/usr/nserve/netspec
/usr/nserve/rfmaster

SEE ALSO

rfmaster(5), **dname**(8), **fumount**(8), **mount**(8), **nlsadmin**(8), **rfstart**(8), **rfstop**(8)

NAME

dump, rdump – incremental file system dump

SYNOPSIS

/usr/etc/dump [*options* [*arguments*]] *filesystem*
/usr/etc/dump [*options* [*arguments*]] *filename* ...

/usr/etc/rdump [*options* [*arguments*]] *filesystem*
/usr/etc/rdump [*options* [*arguments*]] *filename* ...

DESCRIPTION

dump backs up all files in *filesystem*, or files changed after a certain date, or a specified set of files and directories, to magnetic tape, diskettes, or files. *options* is a string that specifies **dump** options, as shown below. Any *arguments* supplied for specific options are given as subsequent words on the command line, in the same order as that of the *options* listed.

If **dump** is called as **rdump,** the dump device defaults to **dumphost:/dev/rmt8.**

If no *options* are given, the default is **9u.**

dump is normally used to back up a complete filesystem. To restrict the dump to a specified set of files and directories on one filesystem, list their names on the command line. In this mode the dump level is set to **0** and the **u** option is ignored.

OPTIONS

0–9 The "dump level." All files in the *filesystem* that have been modified since the last **dump** at a lower dump level are copied to the volume. For instance, if you did a "level **2**" dump on Monday, followed by a "level **4**" dump on Tuesday, a subsequent "level **3**" dump on Wednesday would contain all files modified or added since the "level **2**" (Monday) backup. A "level **0**" dump copies the entire filesystem to the dump volume.

a *archive-file*
Create a dump table-of-contents archive in the specified file, *archive-file*. This file can be used by **restore**(8) to determine whether a file is present on a dump tape, and if so, on which volume it resides. For further information on the use of a dump archive file, see **restore**(8).

b *factor* Blocking factor. Specify the blocking factor for tape writes. The default is 20 blocks per write. Note: the blocking factor is specified in terms of 512 bytes blocks, for compatibility with **tar**(1). The default blocking factor for tapes of density 6250BPI and greater is 64. The default blocking factor for cartridge tapes (**c** option specified) is 126. The highest blocking factor available with most tape drives is 126.

c Cartridge. Use a cartridge instead of the standard half-inch reel. This sets the density to 1000BPI, the blocking factor to 126, and the length to 425 feet. This option also sets the "inter-record gap" to the appropriate length. When cartridge tapes are used, and this option is *not* specified, **dump** will slightly miscompute the size of the tape. If the **b**, **d**, **s** or **t** options are specified with this option, their values will override the defaults set by this option.

d *bpi* Tape density. The density of the tape, expressed in BPI, is taken from *bpi*. This is used to keep a running tab on the amount of tape used per reel. The default density is 1600 except for cartridge tape. Unless a higher density is specified explicitly, **dump** uses its default density — even if the tape drive is capable of higher-density operation (for instance, 6250BPI). Note: the density specified should correspond to the density of the tape device being used, or **dump** will not be able to handle end-of-tape properly. The **d** option is not compatible with the **D** option.

D Diskette. Specify diskette as the dump media.

f *dump-file*
Dump file. Use *dump-file* as the file to dump to, instead of **/dev/rmt8.** If *dump-file* is specified as '–', dump to the standard output. If the file name argument is of the form *machine:device*, dump to a remote machine. Since **dump** is normally run by *root*, the name of the local machine must

appear in the **.rhosts** file of the remote machine. If the file name argument is of the form *user@machine:device*, **dump** will attempt to execute as the specified user on the remote machine. The specified user must have a **.rhosts** file on the remote machine that allows root from the local machine. If **dump** is called as **rdump,** the dump device defaults to **dumphost:/dev/rmt8**. To direct the output to a desired remote machine, set up an alias for **dumphost** in the file **/etc/hosts**.

n Notify. When this option is specified, if **dump** requires attention, it sends a terminal message (similar to **wall**(1)) to all operators in the "operator" group.

s *size* Specify the *size* of the volume being dumped to. When the specified size is reached, **dump** waits for you to change the volume. **dump** interprets the specified size as the length in feet for tapes, and cartridges and as the number of 1024 byte blocks for diskettes. The following are defaults:

tape	2300 feet
cartridge	425 feet
diskette	1422 blocks (Corresponds to a 1.44 Mb diskette, with one cylinder reserved for bad block information.)

t *tracks* Specify the number of tracks for a cartridge tape. On all Sun-2 systems the default is 4 tracks, although some Sun-2 systems have 9 track drives. On all other machines the default is 9 tracks. The **t** option is not compatible with the **D** option.

u Update the dump record. Add an entry to the file **/etc/dumpdates,** for each filesystem successfully dumped that includes the filesystem name, date, and dump level. This file can be edited by the super-user.

v After writing each volume of the dump, the media is rewound and is verified against the filesystem being dumped. If any discrepancies are found, dump will respond as if a write error had occurred; the operator will be asked to mount new media, and dump will attempt to rewrite the volume. Note that *any* change to the filesystem, even the update of the access time on a file will cause the verification to fail. Thus, the verify option can only be used on a quiescent filesystem.

w List the filesystems that need backing up. This information is gleaned from the files **/etc/dumpdates** and **/etc/fstab**. When the **w** option is used, all other options are ignored. After reporting, **dump** exits immediately.

W Like **w**, but includes all filesystems that appear in **/etc/dumpdates,** along with information about their most recent dump dates and levels. Filesystems that need backing up are highlighted.

FILES

/dev/rmt8	default unit to dump to
dumphost:/dev/rmt8	default remote unit to dump to if called as **rdump**
/dev/rst*	Sun386i cartridge tape dump device
/dev/rfd0a	Sun386i 1.44 megabyte 3.5-inch high density diskette drive dump device
/dev/rfdl0a	Sun386i 720 kilobyte 3.5-inch low density diskette drive dump device
/dev/rfd0c	Sun386i 1.44 megabyte 3.5-inch high density diskette drive dump device
/dev/rfdl0c	Sun386i 720 kilobyte 3.5-inch low density diskette drive dump device
/etc/dumpdates	dump date record
/etc/fstab	dump table: file systems and frequency
/etc/group	to find group *operator*
/etc/hosts	

SEE ALSO

bar(1), **fdformat**(1), **tar**(1), **wall**(1), **dump**(5), **fstab**(5), **restore**(8), **shutdown**(8)

DIAGNOSTICS

While running, **dump** emits many verbose messages.

Exit Codes

0 Normal exit.
1 Startup errors encountered.
3 Abort – no checkpoint attempted.

BUGS

Fewer than 32 read errors on the file system are ignored.

Each reel requires a new process, so parent processes for reels already written just hang around until the entire tape is written.

It is recommended that incremental dumps also be performed with the system running in single-user mode.

dump does not support multi-file multi-volume tapes.

NOTES

Operator Intervention

dump requires operator intervention on these conditions: end of volume, end of dump, volume write error, volume open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the **n** option, **dump** interacts with the operator on **dump**'s control terminal at times when **dump** can no longer proceed, or if something is grossly wrong. All questions **dump** poses *must* be answered by typing **yes** or **no**, as appropriate.

Since backing up a disk can involve a lot of time and effort, **dump** checkpoints at the start of each volume. If writing that volume fails for some reason, **dump** will, with operator permission, restart itself from the checkpoint after a defective volume has been replaced.

dump reports periodically, and in verbose fashion. Each report includes estimates of the percentage of the dump completed and how long it will take to complete the dump. The estimated time is given as *hours:minutes*.

Suggested Dump Schedule

It is vital to perform full, "level **0**", dumps at regular intervals. When performing a full dump, bring the machine down to single-user mode using **shutdown**(8). While preparing for a full dump, it is a good idea to clean the tape drive and heads.

Incremental dumps allow for convenient backup and recovery on a more frequent basis of active files, with a minimum of media and time. However there are some tradeoffs. First, the interval between backups should be kept to a minimum (once a day at least). To guard against data loss as a result of a media failure (a rare, but possible occurrence), it is a good idea to capture active files on (at least) two sets of dump volumes. Another consideration is the desire to keep unnecessary duplication of files to a minimum to save both operator time and media storage. A third consideration is the ease with which a particular backed-up version of a file can be located and restored. The following four-week schedule offers a reasonable trade-off between these goals.

	Sun	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Week 1:	**Full**	**5**	**5**	**5**	**5**	**3**
Week 2:		**5**	**5**	**5**	**5**	**3**
Week 3:		**5**	**5**	**5**	**5**	**3**
Week 4:		**5**	**5**	**5**	**5**	**3**

Although the Tuesday — Friday incrementals contain "extra copies" of files from Monday, this scheme assures that any file modified during the week can be recovered from the previous day's incremental dump.

Process Priority of dump

dump uses multiple processes to allow it to read from the disk and write to the media concurrently. Due to the way it synchronizes between these processes, any attempt to run dump with a **nice** (process priority) of '–5' or better will likely make **dump** run *slower* instead of faster.

NAME

dumpfs – dump file system information

SYNOPSIS

/usr/etc/dumpfs *device*

DESCRIPTION

dumpfs prints out the super block and cylinder group information for the file system or special device specified. The listing is very long and detailed. This command is useful mostly for finding out certain file system information such as the file system block size and minimum free space percentage.

SEE ALSO

fs(5), **fsck**(8), **newfs**(8), **tunefs**(8)

NAME

edquota – edit user quotas

SYNOPSIS

/usr/etc/edquota [**–p** *proto-user*] *usernames...*

/usr/etc/edquota –t

DESCRIPTION

edquota is a quota editor. One or more users may be specified on the command line. For each user a temporary file is created with an ASCII representation of the current disk quotas for that user and an editor is then invoked on the file. The quotas may then be modified, new quotas added, etc. Upon leaving the editor, **edquota** reads the temporary file and modifies the binary quota files to reflect the changes made.

The editor invoked is **vi**(1) unless the **EDITOR** environment variable specifies otherwise.

Only the super-user may edit quotas. (In order for quotas to be established on a file system, the root directory of the file system must contain a file, owned by root, called **quotas**. See **quotaon**(8) for details.)

OPTIONS

–p Duplicate the quotas of the prototypical user specified for each user specified. This is the normal mechanism used to initialize quotas for groups of users.

–t Edit the soft time limits for each file system. If the time limits are zero, the default time limits in **<ufs/quota.h>** are used. Time units of sec(onds), min(utes), hour(s), day(s), week(s), and month(s) are understood. Time limits are printed in the greatest possible time unit such that the value is greater than or equal to one.

FILES

quotas	quota file at the file system root
/etc/mtab	mounted file systems

SEE ALSO

quota(1), **vi**(1), **quotactl**(2), **quotacheck**(8), **quotaon**(8), **repquota**(8)

BUGS

The format of the temporary file is inscrutable.

NAME

eeprom – EEPROM display and load utility

SYNOPSIS

eeprom [−] [**−c**] [**−i**] [**−f** *device*] [*field*[=*value*] ...]

SYNOPSIS — SPARCstation 1 SYSTEMS

eeprom [−] [**−f** *device*] [*field*[=*value*] ...]

DESCRIPTION

eeprom displays or changes the values of fields in the EEPROM. It processes fields in the order given. When processing a *field* accompanied by a *value*, **eeprom** makes the indicated alteration to the EEPROM; otherwise it displays the *field*'s value. When given no field specifiers, **eeprom** displays the values of all EEPROM fields. A '−' flag specifies that fields and values are to be read from the standard input (one *field* or *field*=*value* per line).

Only the super-user may alter the EEPROM contents.

eeprom verifies the EEPROM checksums and complains if they are incorrect; if the **−i** flag is specified, erroneous checksums are ignored. If the **−c** flag is specified, all incorrect checksums are recomputed and corrected in the EEPROM.

The PROM monitor supports three security modes designated by the *secure* field: non-secure, command secure, and fully secure.

If *secure*=**none** the PROM monitor runs in the non-secure mode. In this mode all PROM monitor commands are allowed with no password required.

If **secure=command** the PROM monitor is in the command secure mode. In this mode, only the **b** (boot) command with no parameters and the **c** (continue) command with no parameters may be entered without a password being required. Any other command requires that the PROM monitor password be entered.

If **secure=full** the PROM monitor is in the fully secure mode. In this mode, only the **c** (continue) command with no parameters may be entered without a password being required. Entry of any other command requires that the PROM monitor password be entered. Note: the system will not auto-reboot in fully secure mode. The PROM monitor password must be entered before the boot process will take place.

When changing the security mode from non-secure to either command secure or fully secure, **eeprom** prompts for the entry and re-entry of a new PROM password as in the **passwd**(1) command. Changing from one secure mode to the other secure mode, or to the non-secure mode does not prompt for a password. Changing to non-secure mode erases the password.

The content of the **password** field is never displayed to any user. If the security mode is not **none**, the super-user may change the PROM monitor password by entering:

example# eeprom password=

eeprom prompts for a new password to be entered and re-entered.

The field **bad_login** maintains the count of bad login tries. It may be reset to zero (0) by specifying **bad_login=reset**.

OPTIONS

−c	Correct bad checksums. (Ignored on SPARCstation 1 systems.)
−i	Ignore bad checksums. (Ignored on SPARCstation 1 systems.)
−f *device*	Use *device* as the EEPROM device.

FIELDS and VALUES

hwupdate	a valid date (including **today** and **now**)
memsize	8 bit integer (megabytes of memory on machine)
memtest	8 bit integer (megabytes of memory to test)
scrsize	**1024x1024**, **1152x900**, **1600x1280**, or **1440x1440**

watchdog_reboot	**true** or **false**
default_boot	**true** or **false**
bootdev	*charchar*(*hex-int*,*hex-int*,*hex-int*) (with *char* a character, and *hex-int* a hexadecimal integer.)
kbdtype	8 bit integer (0 for all Sun keyboards)
keyclick	**true** or **false**
console	**b&w** or **ttya** or **ttyb** or **color**
custom_logo	**true** or **false**
banner	banner string
diagdev	%c%c (%x,%x,%x) — diagnostic boot device
diagpath	diagnostic boot path
ttya_no_rtsdtr	**true** or **false**
ttyb_no_rtsdtr	**true** or **false**
ttya_use_baud	**true** or **false**
ttyb_use_baud	**true** or **false**
ttya_baud	baud rate (16-bit decimal integer)
ttyb_baud	baud rate (16-bit decimal integer)
columns	number of columns on screen (8-bit integer)
rows	number of rows on screen (8-bit integer)
secure	**none**, **command**, or **full**
bad_login	number of bad login tries (16-bit unsigned integer, 0 if **reset**)
password	PROM monitor password (8-bytes)

FIELDS and VALUES — SPARCstation 1 SYSTEMS

hardware-revision	7 chars (for example, **30Mar88**)
selftest-#megs	32 bit decimal integer (megabytes of memory to test)
watchdog-reboot?	**true** or **false**; **true** to reboot after watchdog reset
boot-from	A string specifying boot string (for example, **le()vmunix**); defaults to **vmunix**
keyboard-click?	**true** or **false**; **true** to enable clicking of keys on each keystroke
input-device	A string specifying one of **keyboard**, **ttya**, or **ttyb**; if the specified device is unavailable, **ttya** is used for both input and output *only* if input-device specified the keyboard *and* output-device specified the screen.
output-device	A string specifying one of **screen**, **ttya**, or **ttyb**; if the specified device is unavailable, **ttya** is used for *both* input and output *only* if input-device specified the keyboard *and* output-device specified the screen.
oem-banner?	**true** or **false**; **true** to use custom banner string instead of Sun banner
oem-banner	80 chars for custom banner string
oem-logo?	**true** or **false**; **true** to display custom logo instead of Sun logo
oem-logo	Name of file (in **iconedit** format) containing custom logo.
boot-from-diag	80 chars specifying diag boot string (for example, **sd()dexec**); defaults to **le()vmunix**
ttya-mode	16 chars to specify 5 comma-separated fields of configuration information (for example, **1200,8,1,n,–**); defaults to **9600,8,1,n,–**. Fields, in left-to-right order, are: baud rate: 110, 300, 1200, 4800, 9600 ... data bits: 5, 6, 7, 8 parity: n(none), e(even), o(odd), m(mark), s(space) stop bits: 1, 1.5, 2 handshake: –(none), h(hardware:rts/cts), s(software:xon/xoff)
ttyb-mode	16 chars to specify 5 comma-separated fields of configuration information (for example, **1200,7,1,n,s**); defaults to **9600,8,1,n,–**.

Fields, in left-to-right order, are:
baud rate: 110, 300, 1200, 4800, 9600...
data bits: 5, 6, 7, 8
stop bits: 1, 1.5, 2
parity: n(none), e(even), o(odd), m(mark), s(space)
handshake: –(none), h(hardware:rts/cts), s(software:xon/xoff)

ttyb-rts-dtr-off **true** or **false**. Defaults to **false**.
ttya-rts-dtr-off **true** or **false**. Defaults to **false**.
ttya-ignore-cd **true** or **false**. Defaults to **true**.
ttyb-ignore-cd **true** or **false**; **true** to ignore the CARRIER DETECT line. Defaults to **true**.
screen-#rows number of rows on output device; defaults to 34 (for some devices actual values used may be less)
screen-#columns number of columns on output device; defaults to 80 (for some devices actual values used may be less)
auto-boot? **true** or **false**; **true** to boot on power-on
scsi-initiator-id An integer between 0 and 7 that specifies the SCSI initiator ID of the onboard SCSI host adapter.
sd-targets An array of 8 integers that map SCSI disk unit numbers to SCSI target numbers. The unit number is used to index into this string. The default settings are **31204567**, which means that unit 0 maps to target 3, unit 1 maps to target 1, and so on.
st-targets An array of 8 integers that map SCSI tape unit numbers to SCSI target numbers. The unit number is used to index into this string. The default settings are **45670123**, which means that unit 0 maps to target 4, unit 1 maps to target 5, and so on.
sunmon-compat? **true** or **false**. Defaults to **true**.
sbus-probe-list Defaults to 0123.
fcode-debug? **true** or **false**. Defaults to **false**.
last-hardware-update Date the CPU board was manufactured or upgraded to the latest hardware revision. The format is a human-readable date string, such as **23May89**.
testarea Defaults to 0.
mfg-switch? **true** or **false**. Defaults to **false**.
diag-switch? **true** or **false**. Defaults to **true**.

FILES

/dev/eeprom

FILES — SPARCstation 1 SYSTEMS

/dev/openprom

SEE ALSO

passwd(1)

PROM User's Manual

NAME

etherd, rpc.etherd – Ethernet statistics server

SYNOPSIS

/usr/etc/rpc.etherd *interface*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

etherd is a server which puts *interface* into promiscuous mode, and keeps summary statistics of all the packets received on that interface. It responds to RPC requests for the summary. You must be root to run **etherd**.

interface is a networking interface such as **ie0**, **ie1**, **ec0**, **ec1** and **le0**.

traffic(1C) displays the information obtained from **etherd** in graphical form.

SEE ALSO

traffic(1C)

NAME

etherfind – find packets on Ethernet

SYNOPSIS

etherfind [**–d**] [**–n**] [**–p**] [**–r**] [**–t**] [**–u**] [**–v**] [**–x**] [**–c** *count*] [**–i** *interface*] [**–l** *length*] *expression*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

etherfind prints out the information about packets on the ethernet that match the boolean *expression*. The short display, without the **–v** option, displays only the destination and src (with port numbers). When an Internet packet is fragmented into more than one ethernet packet, all fragments except the first are marked with an asterisk. With the **–v** option, the display is much more verbose, giving a trace that is suitable for analyzing many network problems. You must be root to invoke **etherfind**.

OPTIONS

–d Print the number of dropped packets. Not necessarily reliable.

–n Do not convert host addresses and port numbers to names.

–p Normally, the selected interface is put into promiscuous mode, so that **etherfind** has access to all packets on the ethernet. However, when the **–p** flag is used, the interface will not go promiscuous.

–r RPC mode: treat each packet as an RPC message, printing the program and procedure numbers. Routing packets are also more fully decoded using this option, and Network Interface Service (NIS) and NFS requests have their arguments printed.

–t Timestamps: precede each packet listing with a time value in seconds and hundredths of seconds since the first packet.

–u Make the output line buffered.

–v Verbose mode: print out some of the fields of TCP and UDP packets.

–x Dump the packet in hex, in addition to the line printed for each packet by default. Use the **–l** option to limit this printout.

–c *count*

Exit after receiving *count* packets. This is sometimes useful for dumping a sample of ethernet traffic to a file for later analysis.

–i *interface*

etherfind listens on *interface*. The program **netstat**(8C) when invoked with the **–i** flag lists all the interfaces that a machine has.

–l *length*

Use with the **–x** option to limit the number of bytes printed out.

expression

The syntax of *expression* is similar to that used by **find**(1). Here are the allowable primaries.

dst *destination*

True if the destination field of the packet is *destination*, which may be either an address or a name.

src *source*

True if the source field of the packet is *source*, which may be either an address or a name.

host *name*
True if either the source or the destination of the packet is *name*.

between *host1 host2*
True if either the source of the packet is *host1* and the destination *host2*, or the source is *host2* and the destination *host1*.

dstnet *destination*
True if the destination field of the packet has a network part of *destination*, which may be either an address or a name.

srcnet *source*
True if the source field of the packet has a network part of *source*, which may be either an address or a name.

srcport *port*
True if the packet has a source port value of *port*. This will check the source port value of either UDP or TCP packets (see **tcp**(4P)), and **udp**(4P)). The *port* can be a number or a name used in **/etc/services**.

dstport *port*
True if the packet has a destination port value of *port*. The *port* can be a number or a name.

less *length*
True if the packet has a length less than or equal to *length*.

greater *length*
True if the packet has a length greater than or equal to *length*.

–proto *protocol*
True if the packet is an IP packet (see **ip**(4P)) of protocol type *protocol*. *Protocol* can be a number or one of the names **icmp**, **udp**, **nd**, or **tcp**.

byte *byte op value*
True if byte number *byte* of the packet is in relation *op* to *value*. Legal values for *op* are **+**, **<**, **>**, **&**, and **|**. Thus **4=6** is true if the fourth byte of the packet has the value 6, and **20&0xf** is true if byte twenty has one of its four low order bits nonzero.

broadcast
True if the packet is a broadcast packet.

arp True if the packet is an ARP packet (see **arp**(4P)).

rarp True if the packet is a rarp packet.

–ip True if the packet is an IP packet.

–decnet
True if the packet is a DECNET packet.

–apple True if the packet is an AppleTalk protocol packet.

The primaries may be combined using the following operators (in order of decreasing precedence):

A parenthesized group of primaries and operators (parentheses are special to the Shell and must be escaped).

The negation of a primary ('**not**' is the unary *not* operator).

Concatenation of primaries (the *and* operation is implied by the juxtaposition of two primaries, or can be specified with **'and'**).

Alternation of primaries (**'or'** is the *or* operator).

EXAMPLE

To find all packets arriving at or departing from the host **sundown**, or that are ICMP packets:

example% etherfind host sundown or proto icmp

SEE ALSO

find(1), **traffic**(1C), **arp**(4P), **ip**(4P), **nit**(4P) **tcp**(4P), **udp**(4P), **netstat**(8C)

BUGS

The syntax is painful.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

exportfs – export and unexport directories to NFS clients

SYNOPSIS

/usr/etc/exportfs [**–aiuv**] [**–o** *options*] [*pathname*]

DESCRIPTION

exportfs makes a local directory or filename available for mounting over the network by NFS clients. It is normally invoked at boot time by the **/etc/rc.local** script, and uses information contained in the **/etc/exports** file to export *pathname* (which must be specified as a full pathname). The super-user can run **exportfs** at any time to alter the list or characteristics of exported directories and filenames. Directories and files that are currently exported are listed in the file **/etc/xtab**.

With no options or arguments, **exportfs** prints out the list of directories and filenames currently exported.

OPTIONS

–a All. Export all pathnames listed in **/etc/exports**, or if **–u** is specified, unexport all of the currently exported pathnames.

–i Ignore the options in **/etc/exports.** Normally, **exportfs** will consult **/etc/exports** for the options associated with the exported pathname.

–u Unexport the indicated pathnames.

–v Verbose. Print each directory or filename as it is exported or unexported.

–o *options*

Specify a comma-separated list of optional characteristics for the pathname being exported. *options* can be selected from among:

ro Export the pathname read-only. If not specified, the pathname is exported read-write.

rw=*hostname*[**:***hostname*]...

Export the pathname read-mostly. Read-mostly means exported read-only to most machines, but read-write to those specified. If not specified, the pathname is exported read-write to all.

anon=*uid*

If a request comes from an unknown user, use UID as the effective user ID. Note: root users (UID 0) are always considered "unknown" by the NFS server, unless they are included in the **root** option below. The default value for this option is –2. Setting the value of "anon" to –1 disables anonymous access. Note: by default secure NFS accepts insecure requests as anonymous, and those wishing for extra security can disable this feature by setting "anon" to –1.

root=*hostname*[**:***hostname*]...

Give root access only to the root users from a specified *hostname*. The default is for no hosts to be granted root access.

access=*client*[**:***client*]...

Give mount access to each *client* listed. A *client* can either be a hostname, or a netgroup (see **netgroup**(5)). Each *client* in the list is first checked for in the **/etc/netgroup** database, and then the **/etc/hosts** database. The default value allows any machine to mount the given directory.

secure Require clients to use a more secure protocol when accessing the directory.

FILES

/etc/exports	static export information
/etc/xtab	current state of exported pathnames
/etc/netgroup	

SEE ALSO

exports(5), **netgroup**(5), **showmount**(8)

WARNINGS

You cannot export a directory that is either a parent- or a sub-directory of one that is currently exported and *within the same filesystem*. It would be illegal, for example, to export both **/usr** and **/usr/local** if both directories resided in the same disk partition.

NAME

extract_patch – extract and execute patch files from installation tapes

SYNOPSIS

extract_patch [**–d***device* [**–r***remote-host*]] [**–p***patch-name*] [**–DEFAULT**]

DESCRIPTION

extract_patch extracts a patch from a release tape onto the current system. If no options are specifed, it prompts for input as to the patch name, tape device, or remote hostname from which to the software is to be installed. If the named patch cannot be found, a list of valid patches are printed.

If the named patch is found then the patch is extracted from the tape onto the system. If there is a **README** file in the extracted contents then the user is given a chance to view it. If there is a patch installation program the user is given a chance to run it.

Patches must appear in the tape's table of contents, and must have a name that starts with "Patch_".

OPTIONS

–d*device*

Install from the indicated tape drive, such as **st0**, or **mt0**.

–r*remote-host*

Install from the device given in the **–d** option on the indicated remote host.

–p*patch-name*

Specifes the name of the patch to extract.

–DEFAULT

Execute the installation script using all default values. Otherwise the installation script prompts for any optional values.

SEE ALSO

extract_unbundled(8)

NAME

extract_unbundled – extract and execute unbundled-product installation scripts

SYNOPSIS

extract_unbundled [–**d***device* [–**r***remote-host*]] [–**DEFAULT**]

DESCRIPTION

extract_unbundled extracts and executes the installation scripts from release tapes for Sun unbundled software products. If no options are specified, it prompts for input as to the tape device, or remote hostname from which to the software is to be installed. For information about installing a specific product, refer to the installation manual that accompanies that product.

OPTIONS

–**d***device*

Install from the indicated tape drive, such as **st0** or **mt0**.

–**r***remote_host*

Install from the device given in the –**d** option on the indicated remote host.

–**DEFAULT**

Execute the installation script using all default values. Otherwise the installation script prompts for any optional values.

NAME

fastboot, fasthalt – reboot/halt the system without checking the disks

SYNOPSIS

/usr/etc/fastboot [*boot-options*]

/usr/etc/fasthalt [*halt-options*]

DESCRIPTION

fastboot and **fasthalt** are shell scripts that reboot and halt the system without checking the file systems. This is done by creating a file **/fastboot**, then invoking the **reboot**(8) program. The system startup script, **/etc/rc**, looks for this file and, if present, skips the normal invocation of **fsck**(8).

FILES

/usr/etc/fastboot
/etc/rc

SEE ALSO

fsck(8), **halt**(8), **init**(8), **rc**(8), **reboot**(8)

NAME

fingerd, in.fingerd – remote user information server

SYNOPSIS

/usr/etc/in.fingerd

DESCRIPTION

fingerd implements the server side of the Name/Finger protocol, specified in RFC 742. The Name/Finger protocol provides a remote interface to programs which display information on system status and individual users. The protocol imposes little structure on the format of the exchange between client and server. The client provides a single "command line" to the finger server which returns a printable reply.

fingerd waits for connections on TCP port 79. Once connected it reads a single command line terminated by a LINEFEED which is passed to **finger**(1). **fingerd** closes its connections as soon as the output is finished.

If the line is null (only a LINEFEED is sent) then **finger** returns a "default" report that lists all people logged into the system at that moment.

If a user name is specified (for instance, ericLINEFEED) then the response lists more extended information for only that particular user, whether logged in or not. Allowable "names" in the command line include both "login names" and "user names". If a name is ambiguous, all possible derivations are returned.

SEE ALSO

finger(1)

Harrenstien, Ken, *NAME/FINGER*, RFC 742, Network Information Center, SRI International, Menlo Park, Calif., December 1977.

BUGS

Connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation. **fingerd** should be taught to filter out IAC's and perhaps even respond negatively (IAC *will not)* to all option commands received.

NAME

fonftlip – create Sun386i-style vfont file

SYNOPSIS

fontflip fontname [**–o** *newfontname*]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0*.x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

fontflip takes as input a vfont file (Sun-3 fixedwidthfont) and creates a Sun386i system vfont. This new font is a bitflipped version of its input. The new font is named *oldfont*.**flip** unless otherwise specified.

OPTIONS

–o *newfontname* Specify the name of the new flipped font.

FILES

/usr/lib/fonts/fixedwidthfonts

SEE ALSO

vfont(5)

NAME

format – disk partitioning and maintenance utility

SYNOPSIS

format [**–f** *command-file*] [**–l** *log-file*] [**–x** *data-file*] [**–d** *disk-name*] [**–t** *disk_type*] [**–p** *partition-name*] [**–s**] *diskname*...

DESCRIPTION

format enables you to format, label, repair and analyze disks on your Sun computer. Unlike previous disk maintenance programs, **format** runs under SunOS. Because there are limitations to what can be done to the system disk while the system is running, **format** is also supported within the memory-resident system environment. For most applications, however, running **format** under SunOS is the more convenient approach.

If no *disk-list* is present, **format** uses the disk list defined in the data file specified with the **–x** option. If that option is omitted, the data file defaults to **format.dat** in the current directory, or else **/etc/format.dat**.

OPTIONS

–f *command-file*

Take command input from *command-file* rather than the standard input. The file must contain commands that appear just as they would if they had been entered from the keyboard. With this option, **format** does not issue **continue?** prompts.

–l *log-file*

Log a transcript of the **format** session to the indicated *log-file*, including the standard input, the standard output and the standard error.

–x *data-file*

Use the disk list contained in *data-file*.

–d *disk_name*

Specify which disk should be made current upon entry into the program. The disk is specified by its logical name (for instance, - xy0). This can also be accomplished by specifying a single disk in the disk list.

–t *disk-type*

Specify the type of disk which is current upon entry into the program, A disk's type is specified by name in the data file. This option can only be used if a disk is being made current as described above.

–p *partition-name*

Specify the partition table for the disk which is current upon entry into the program. The table is specified by its name as defined in the data file. This option can only be used if a disk is being made current, and its type is either specified or available from the disk label.

–s Silent. Suppress all of the standard output. Error messages are still displayed. This is generally used in conjunction with the **–f** option.

FILES

/etc/format.dat default data file

SEE ALSO

System and Network Administration

NAME

fpa_download – download to the Floating Point Accelerator

SYNOPSIS

fpa_download [**–d**] [**–r**] [**–v**] [**–u** *ufile*] [**–m** *mfile*] [**–c** *cfile*]

AVAILABILITY

fpa_download applies to Sun-3 and Sun-3x systems equipped with either an FPA or FPA+.

DESCRIPTION

fpa_download writes microcode, map, and constants files to FPA and FPA+ boards. FPA requires a map file; FPA+ does not.

Root execution level is required to download (d,u,m and c options). **fpa_download** is called from **/etc/rc.local** when **/dev/fpa** exists.

Given no arguments, **fpa_download** prints whether an FPA, or FPA+ is installed.

OPTIONS

–d	Download microcode, constants, and map files. Enable default file names.
–r	Print microcode and constant revision.
–v	Verbose mode.
–u *ufile*	Download microcode from *ufile*.
–m *mfile*	Download map from *mfile* (FPA only).
–c *cfile*	Download constants from *cfile*.

FILES

/dev/fpa	device file for both FPA and FPA+.
/usr/etc/fpa/fpa_micro_bin	default microcode file (*ufile*) for FPA.
/usr/etc/fpa/fpa_constants	default constants file (cfile) for FPA
/usr/etc/fpa/fpa_micro_map	default map file (mfile) for FPA
/usr/etc/fpa/fpa_micro_bin+	default microcode file (ufile) for FPA+
/usr/etc/fpa/fpa_constants+	default constants file (cfile) for FPA+

SEE ALSO

fpa(4)

DIAGNOSTICS

The following diagnostics are printed when **fpa_download** encounters a serious error and asks the kernel to disable the FPA. This might occur if the microcode, map, or constants files are corrupted, or if there is an FPA or system hardware problem.

FPA Download Failed - FPA ioctl failed

An ioctl() on **/dev/fpa** failed, possibly due to a hung FPA pipe.

FPA Failed Download - FPA Bus Error

Received a SIGFPE.

FPA Failed Download - Upload mismatch

After each file is written to the FPA/FPA+, **fpa_download** uploads the contents of FPA memory and compares it with the source. They should always match.

NAME

fparel – Sun FPA online reliability tests

SYNOPSIS

fparel [**–p***n*] [**–v**]

AVAILABILITY

Not available on Sun386i systems.

DESCRIPTION

fparel is a command to execute the Sun FPA online confidence and reliability test program. **fparel** tests about 90% of the functions of the FPA board, and tests all FPA contexts not in use by other processes. **fparel** runs without disturbing other processes that may be using the FPA. **fparel** can only be run by the super-user.

After a successful pass, **fparel** writes

time, date: Sun FPA Passed. The contexts tested are: 0, 1, ... 31

to the file **/var/adm/diaglog**.

If a pass fails, **fparel** writes

time, date: Sun FPA failed

along with the test name and context number that failed, to the file **/var/adm/diaglog**. **fparel** then broadcasts the message

time, date: Sun FPA failed, disabled, service required

to all users of the system. Next, **fparel** causes the kernel to disable the FPA. Once the kernel disables the FPA, the system must be rebooted to make it accessible.

The file **/etc/rc.local** should contain an entry to cause **fparel** to be invoked upon reboot to be sure that the FPA remains unaccessible in cases where rebooting doesn't correct the problem. See **rc**(8).

The **crontab**(5) file for root should contain an entry indicating that **cron**(8) is to run **fparel** daily, such as:

7 2 * * * /usr/etc/fpa/fparel

which causes **fparel** to run at seven minutes past two, every day. See **cron**(8) and **crontab**(5) for details.

OPTIONS

–p*n* Perform *n* passes. Default is *n*=1. **–p0** means perform 2147483647 passes.

–v Run in verbose mode with detailed test results to the standard output.

FILES

/var/adm/diaglog Log of **fparel** diagnostics.
/etc/rc.local
/var/spool/cron/crontabs/root
/usr/etc/fpa/* directory containing FPA microcode, data files, and loader

SEE ALSO

crontab(5), **cron**(8), **fpaversion**(8), **rc**(8)

NAME

fpaversion – print FPA version, load microcode

SYNOPSIS

fpaversion [**–chlqv**] [**–t** [**cdhimprstvxCIMS**]]

AVAILABILITY

Available only on Sun-3 and Sun-3x systems equipped with either an FPA or an FPA+.

DESCRIPTION

fpaversion performs various tests on the FPA or FPA+. Without arguments, it prints the microcode version number and constants currently installed on **/dev/fpa**. **fpaversion** also performs a quick test to ensure proper operation and reports whether an FPA or an FPA+ is installed.

OPTIONS

–c Continue tests after an error.

–h Help. Print command-line summary.

–l Loop through tests infinitely.

–q Quiet output. Print out only error messages.

–v Verbose output.

–t Specify certain tests:

- **c** Command register format instructions.
- **d** Double precision format instructions.
- **h** Help. Print summary of test specifiers.
- **i** Imask register.
- **m** Mode register.
- **p** Simple pipe sequencing.
- **r** User registers for all contexts.
- **s** Single precision format instructions.
- **t** Status generation.
- **v** Print version number and date of microcode, and constants. Report whether an FPA or an FPA+ is installed.
- **x** Extended format instructions.
- **C** Check checksum for microcode, mapping RAM, and constant RAM for the FPA. Check checksum for microcode RAM and constant RAM for the FPA+.
- **I** Allows interactive reads and writes to the FPA.
- **M** Command register format matrix instructions.
- **S** Shadow registers.

FILES

/dev/fpa	physical FPA device
/usr/etc/fpa/fpa_micro_bin	microcode binaries for the FPA
/usr/etc/fpa/fpa_micro_map	microcode map binaries for the FPA
/usr/etc/fpa/fpa_constants	microcode data file for the FPA
/usr/etc/fpa/fpa_micro_bin+	microcode binaries for the FPA+
/usr/etc/fpa/fpa_constants+	microcode data file for the FPA+
/usr/etc/fpa/fpa_download	microcode loader

SEE ALSO

fpa_download(8), **fparel**(8), **sundiag**(8)

DIAGNOSTICS

If a test fails, its name, along with the actual and expected results will be printed.

NAME

fpurel – perform tests the Sun Floating Point Co-processor.

SYNOPSIS

fpurel [**–v**] [**–p**[*count*]] [**–r**]

DESCRIPTION

fpurel performs a series of functional and computational tests for the Sun Floating Point Co-processor to verify that it is operational and accurate. With no options, **fpurel** runs one pass silently in the foreground and only reports errors if any are found.

OPTIONS

–v Verbose. Display the name and results of each test on the console. The default is to run silently.

–p[*count*]
Passcount. Specify the number of times to run the test suite. The default is to run one pass.

–r Disable stop on error. Continue to run if errors are detected. The default is to display the error message and to stop testing when an error is detected.

EXAMPLE

This example uses **fpurel** from the **/usr/diag** directory. If no errors are detected, then no information is displayed.

```
% /usr/diag/fpurel
```

NAME

fpuversion4 – print the Sun-4 FPU version

SYNOPSIS

/usr/etc/fpuversion4

AVAILABILITY

Sun-4 systems only.

DESCRIPTION

fpuversion4 reads the **%fsr** register to determine the FPU version installed on a Sun-4. The printed version field contains a value in the range 0-7; by SPARC convention 7 indicates that no FPU is installed, so floating-point instructions are always emulated in the kernel.

NAME

fsck – file system consistency check and interactive repair

SYNOPSIS

/usr/etc/fsck **–p** [*filesystem* ...]

/usr/etc/fsck [**–b** *block#*] [**–w**] [**–y**] [**–n**] [**–c**] [*filesystem*] ...

DESCRIPTION

The first form of **fsck** preens a standard set of file systems or the specified file systems. It is normally used in the **/etc/rc** script during automatic reboot. In this case, **fsck** reads the table **/etc/fstab** to determine the file systems to check. It inspects disks in parallel, taking maximum advantage of I/O overlap to check the file systems as quickly as possible.

Normally, the root file system is checked in pass 1; other root-partition file systems are checked in pass 2. Small file systems on separate partitions are checked in pass 3, while larger ones are checked in passes 4 and 5.

Only partitions marked in **/etc/fstab** with a file system type of "4.2" and a non-zero pass number are checked.

fsck corrects innocuous inconsistencies such as: unreferenced inodes, too-large link counts in inodes, missing blocks in the free list, blocks appearing in the free list and also in files, or incorrect counts in the super block, automatically. It displays a message for each inconsistency corrected that identifies the nature of, and file system on which, the correction is to take place. After successfully correcting a file system, **fsck** prints the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

If **fsck** encounters other inconsistencies that it cannot fix automatically, it exits with an abnormal return status (and the reboot fails).

If sent a QUIT signal, **fsck** will finish the file system checks, then exit with an abnormal return status that causes the automatic reboot to fail. This is useful when you wish to finish the file system checks, but do not want the machine to come up multiuser.

Without the **–p** option, **fsck** audits and interactively repairs inconsistent conditions on file systems. In this case, it asks for confirmation before attempting any corrections. Inconsistencies other than those mentioned above can often result in some loss of data. The amount and severity of data lost can be determined from the diagnostic output.

The default action for each correction is to wait for the operator to respond either **yes** or **no**. If the operator does not have write permission on the file system, **fsck** will default to a **–n** (no corrections) action.

If no file systems are given to **fsck** then a default list of file systems is read from the file **/etc/fstab**.

Inconsistencies checked in order are as follows:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode or the free list outside the range of the file system.
- Incorrect link counts.
- Incorrect directory sizes.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks, file pointing to unallocated inode, inode number out of range.
- Super Block checks: more blocks for inodes than there are in the file system.
- Bad free block list format.
- Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. If the **lost+found** directory does not exist, it is created. If there is insufficient space its size is increased.

A file system may be specified by giving the name of the cooked or raw device on which it resides, or by giving the name of its mount point. If the latter is given, **fsck** finds the name of the device on which the file system resides by looking in **/etc/fstab**.

Checking the raw device is almost always faster.

OPTIONS

–b Use the block specified immediately after the flag as the super block for the file system. Block 32 is always an alternate super block.

–w Check writable file systems only.

–y Assume a **yes** response to all questions asked by **fsck**; this should be used with extreme caution, as it is a free license to continue, even after severe problems are encountered.

–n Assume a **no** response to all questions asked by **fsck**; do not open the file system for writing.

–c If the file system is in the old (static table) format, convert it to the new (dynamic table) format. If the file system is in the new format, convert it to the old format provided the old format can support the filesystem configuration. In interactive mode, **fsck** will list the direction the conversion is to be made and ask whether the conversion should be done. If a negative answer is given, no further operations are done on the filesystem. In preen mode, the direction of the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the file systems are being converted at once. The format of a file system can be determined from the first line of output from **dumpfs**(8)

FILES

/etc/fstab default list of file systems to check

DIAGNOSTICS

The diagnostics produced by **fsck** are fully enumerated and explained in *System and Network Administration.*

EXIT STATUS

0 Either no errors detected or all errors were corrected.

4 Root file system errors were corrected. The system must be rebooted.

8 Some uncorrected errors exist on one or more of the file systems checked, there was a syntax error, or some other operational error occurred.

12 A signal was caught during processing.

SEE ALSO

fs(5), **fstab**(5), **dumpfs**(8), **newfs**(8), **mkfs**(8), **panic**(8S), **reboot**(8), **rexecd**(8C), **ypserv**(8)

System and Network Administration

BUGS

There should be some way to start a '**fsck –p**' at pass *n*.

NAME

fsirand – install random inode generation numbers

SYNOPSIS

fsirand [**–p**] *special*

DESCRIPTION

fsirand installs random inode generation numbers on all the inodes on device *special*, and also installs a filesystem ID in the superblock. This helps increase the security of filesystems exported by NFS.

fsirand must be used only on an unmounted filesystem that has been checked with **fsck**(8). The only exception is that it can be used on the root filesystem in single-user mode, if the system is immediately rebooted afterwords.

OPTIONS

–p Print out the generation numbers for all the inodes, but do not change the generation numbers.

SEE ALSO

fsck(8)

NAME

ftpd, in.ftpd – TCP/IP Internet File Transfer Protocol server

SYNOPSIS

/usr/etc/in.ftpd [**–dl**] [**–t***timeout*] *host.socket*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

ftpd is the TCP/IP Internet File Transfer Protocol (FTP) server process. The server is invoked by the Internet daemon **inetd**(8C) each time a connection to the FTP service (see **services**(5)) is made, with the connection available as descriptor 0 and the host and socket the connection originated from (in hex and decimal respectively) as argument.

Inactive connections are timed out after 60 seconds.

If the **–d** option is specified, debugging information is logged to the system log daemon, **syslogd**(8).

If the **–l** option is specified, each FTP session is logged to **syslogd**.

The FTP server will timeout an inactive session after 15 minutes. If the **–t** option is specified, the inactivity timeout period will be set to *timeout*.

The FTP server currently supports the following FTP requests; case is not distinguished.

Request	Description
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (**ls –lg**)
MKD	make a directory
MODE	specify data transfer *mode*
NLST	give name list of files in directory (**ls**)
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name

STOR store a file

STOU store a file with a unique name

STRU specify data transfer *structure*

TYPE specify data transfer *type*

USER specify user name

XCUP change to parent of current working directory

XCWD change working directory

XMKD make a directory

XPWD print the current working directory

XRMD remove a directory

The remaining FTP requests specified in RFC 959 are recognized, but not implemented.

The FTP server will abort an active file transfer only when the **ABOR** command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in RFC 959.

ftpd interprets file names according to the "globbing" conventions used by **csh**(1). This allows users to utilize the metacharacters '*** ? [] {}~**'.

ftpd authenticates users according to three rules.

- The user name must be in the password data base, **/etc/passwd**, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.
- If the file **/etc/ftpusers** exists, the user name must not appear in that file.
- The user must have a standard shell returned by **getusershell**(3).
- If the user name is "anonymous" or "ftp", an anonymous FTP account must be present in the password file (user "ftp"). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

In the last case, **ftpd** takes special measures to restrict the client's access privileges. The server performs a **chroot**(2) command to the home directory of the "ftp" user. In order that system security is not breached, it is recommended that the "ftp" subtree be constructed with care; the following rules are recommended.

~ftp Make the home directory owned by "ftp" and unwritable by anyone.

~ftp/bin Make this directory owned by the super-user and unwritable by anyone. The program **ls**(1V) must be present to support the list commands. This program should have mode 111. Since the default /bin/ls command is linked with a shared library, so you need to set up the files for dynamic linking as well.

~ftp/usr/lib/ld.so
the runtime loader must be present and executable.

~ftp/dev/zero
used by the runtime loader, create this with the command "mknod zero c 3 12".

~ftp/usr/lib/libc.so.*
should be a copy of the latest version of the shared C library.

~ftp/etc Make this directory owned by the super-user and unwritable by anyone. The files **passwd**(5) and **group**(5) must be present for the **ls** command to work properly. These files should be mode 444.

~ftp/pub Make this directory mode 777 and owned by "ftp". Users should then place files which are to be accessible via the anonymous account in this directory.

DIAGNOSTICS

ftpd logs various errors to the system log daemon, **syslogd**, with a facility code of **daemon**. The messages are listed here, grouped by severity level.

Err Severity

getpeername failed: *reason*
A **getpeername**(2) call failed.

getsockname failed: *reason*
A **getsockname**(2) call failed.

signal failed: *reason*
A **signal (3V)** (see **signal**(3V)) call failed.

setsockopt failed: *reason*
A **setsockopt** call (see **getsockopt**(2)) failed.

ioctl failed: *reason*
A **ioctl**(2) call failed.

directory: *reason*
ftpd did not have write permission on the directory *directory* in which a file was to be created by the **STOU** command.

Info Severity

These messages are logged only if the **–l** flag is specified.

FTPD: connection from *host* **at** *time*
A connection was made to **ftpd** from the host *host* at the date and time *time*.

FTPD: User *user* **timed out after** *timeout* **seconds at** *time*
The user *user* was logged out because they hadn't entered any commands after *timeout* seconds; the logout occurred at the date and time *time*.

Debug Severity

These messages are logged only if the **–d** flag is specified.

TPD: command: *command*
A command line containing *command* was read from the FTP client.

lost connection
The FTP client dropped the connection.

<--- replycode
<--- replycode–
A reply was sent to the FTP client with the reply code *replycode*. The next message logged will include the message associated with the reply. If a – follows the reply code, the reply is continued on later lines.

SEE ALSO

csh(1), **ftp**(1C), **ls**(1V), **chroot**(2) **getpeername**(2), **getsockname**(2), **getsockopt**(2), **ioctl**(2), **getusershell**(3), **ftpusers**(5), **group**(5), **passwd**(5), **services**(5), **inetd**(8C), **syslogd**(8)

Postel, Jon, and Joyce Reynolds, *File Transfer Protocol (FTP)*, RFC 959, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

BUGS

The anonymous account is inherently dangerous and should be avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user ID of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

NAME

fumount – force unmount of an advertised RFS resource

SYNOPSIS

fumount [**–w** *seconds*] *resource*

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

fumount unadvertises *resource* and disconnects remote access to the resource.

When the forced unmount occurs, an administrative shell script, **rfuadmin**, is started on each remote system that has the resource mounted If a grace period is specified (in seconds), **rfuadmin**(8) is started with the **fuwarn** option. When the actual forced unmount is ready to occur, **rfuadmin**(8) is started with the **fumount** option. See **rfuadmin**(8) for information on the action taken in response to the forced unmount.

This command is restricted to the super-user.

An error message will be sent to standard error if any of the following are true of *resource*:

- It does not physically reside on the local machine.
- It is an invalid resource name.
- It is not currently advertised and is not remotely mounted.

OPTION

–w *seconds* Delay execution of the disconnect *seconds* seconds.

SEE ALSO

adv(8), **mount**(8), **rfuadmin**(8), **rfudaemon**(8), **unadv**(8)

NAME

fusage – RFS disk access profiler

SYNOPSIS

fusage [[*mount_point*] | [*advertised_resource*] | [*block_special_device*] [. . .]]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

When used with no options, **fusage** reports block I/O transfers, in kilobytes, to and from all locally mounted file systems and advertised Remote File Sharing resources on a per client basis. The count data are cumulative since the time of the mount. When used with an option, **fusage** reports on the named file system, advertised resource, or block special device.

The report includes one section for each file system and advertised resource and has one entry for each machine that has the directory remotely mounted, ordered by decreasing usage. Sections are ordered by device name; advertised resources that are not complete file systems will immediately follow the sections for the file systems they are in.

SEE ALSO

df(1V), **adv**(8), **crash**(8), **mount**(8)

NAME

fuser – identify processes using a file or file structure

SYNOPSIS

/usr/etc/fuser [**–ku**] *filename* | *resource* [–] [[**–ku**] *filename* | *resource*]

DESCRIPTION

fuser outputs the process IDs of the processes that are using the *filename*s or remote *resources* specified as arguments. Each process ID is followed by a letter code. Possible code letters and an explanation of how the process is using the file are given below:

- **c** its current directory
- **p** the parent of its current directory (only when the file is being used by the system)
- **r** its root directory
- **v** process has **exec**'ed or **mmap**'ed file

For block special devices with mounted file systems, all processes using any file on that device are listed. For remote resource names, all processes using any file associated with that remote resource are reported. **fuser** cannot use the mount point of the remote resource to report all processes using any file associated with that remote resource; it must use the resource name. For all other types of files (text files, executables, directories, devices, etc.) only the processes using that file are reported.

The process IDs are printed as a single line on the standard output, separated by SPACE characters and terminated with a single NEWLINE. All other output is written on standard error.

Any user with permission to read **/dev/kmem** and **/dev/mem** can use **fuser**.

Only the super-user can terminate another user's process

OPTIONS

If more than one group of files are specified, the options may be respecified for each additional group of files.

- **–** Cancel the options currently in force. The new set of options applies to the next group of files.
- **–k** Send SIGKILL signal to each process. Since this option spawns kills for each process, the kill messages may not show up immediately (see **kill**(2V)).
- **–u** User login name, in parentheses, also follows the process ID.

FILES

/vmunix	system namelist
/dev/kmem	system image
/dev/mem	system image

SEE ALSO

ps(1), **kill**(2V), **signal**(3V), **mount**(8)

NAME

fwtmp, wtmpfix – manipulate connect accounting records

SYNOPSIS

/usr/lib/acct/fwtmp [**–ci**]

/usr/lib/acct/wtmpfix [*filename* ...]

DESCRIPTION

fwtmp

fwtmp reads from the standard input and writes to the standard output, converting binary records of the type found in **wtmp** to formatted ASCII records. The ASCII version is useful to enable editing bad records, using a text editor, or general purpose maintenance of the file.

wtmpfix

wtmpfix examines the standard input or named files in **wtmp** format, corrects the time/date stamps to make the entries consistent, and writes to the standard output. A '–' can be used in place of *filename* to indicate the standard input. If time/date corrections are not performed, **acctcon1** fails when it encounters certain date-change records.

Each time the date is set, a pair of date change records are written to **/var/adm/wtmp**. The first record is the old date denoted by the string '|' placed in the line field of the **<utmp.h>** structure. The second record specifies the new date and is denoted by the string '{' placed in the line field. **wtmpfix** uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, **wtmpfix** checks the validity of the name field to ensure that it consists solely of alphanumeric characters or SPACE characters. If it encounters a name that is considered invalid, it changes the login name to **INVALID** and writes a diagnostic message to the standard error. In this way, **wtmpfix** reduces the chance that **acctcon1** will fail when processing connect accounting records.

OPTIONS

fwtmp

–c Write output in binary form.

–i Input is in ASCII form.

FILES

/var/adm/wtmp

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **runacct**(8)

NAME

gettable – get DARPA Internet format host table from a host

SYNOPSIS

/usr/etc/gettable *host*

DESCRIPTION

gettable is a simple program used to obtain the DARPA Internet host table from a "hostname" server. The indicated *host* is queried for the table. The table, if retrieved, is placed in the file **hosts.txt**.

gettable operates by opening a TCP connection to the port indicated in the service specification for "hostname" . A request is then made for "ALL" names and the resultant information is placed in the output file.

gettable is best used in conjunction with the **htable**(8) program which converts the DARPA Internet host table format to that used by the network library lookup routines.

SEE ALSO

intro(3), **htable**(8)

Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, *HOSTNAME Server*, RFC 953, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

BUGS

Should allow requests for only part of the database.

NAME

getty – set terminal mode

SYNOPSIS

/usr/etc/getty [*type* [*tty*]]

Sun386i SYSTEM SYNOPSIS

/usr/etc/getty [**–n**] [*type* [*tty*]]

DESCRIPTION

getty, which is invoked by **init**(8), opens and initializes a tty line, reads a login name, and invokes **login**(1).

The *tty* argument is the name of the character-special file in **/dev** that corresponds to the terminal. If there is no *tty* argument, or the argument is '–', the tty line is assumed to be opened as file descriptor **0**.

The *type* argument, if supplied, is used as an index into the **gettytab**(5) database—to determine the characteristics of the line. If this argument is absent, or if there is no such entry, the default entry is used. If there is no **/etc/gettytab** file, a set of system-supplied defaults is used.

When the indicated entry is located, **getty** clears the terminal screen, prints a banner heading, and prompts for a login name. Usually, either the banner or the login prompt includes the system's hostname.

Next, **getty** prompts for a login and reads the login name, one character at a time. When it receives a null character (which is assumed to be the result pressing the BREAK , or "interrupt" key), **getty** switches to the entry **gettytab** entry named in the **nx** field. It reinitializes the line to the new characteristics, and then prompts for a login once again. This mechanism typically is used to cycle through a set of line speeds (baud rates) for each terminal line. For instance, a rotary dialup might have entries for the speeds: 300, 1200, 150, and 110 baud, with each **nx** field pointing to the next one in succession.

The user terminates login input line with a NEWLINE or RETURN character. The latter is preferable; it sets up the proper treatment of RETURN characters (see **tty**(4)). **getty** checks to see if the terminal has only upper-case alphabetical characters. If all alphabetical characters in the login name are in upper case, the system maps them along with all subsequent upper-case input characters to lower-case internally; they are displayed in upper case for the benefit of the terminal. To force recognition of an upper-case character, the shell allows them to be quoted (typically by preceding each with a backslash, '\').

Finally, **getty** calls **login**(1) with the login name as an argument.

getty can be set to time out after a certain interval; this hangs up dial-up lines if the login name is not entered in time.

Sun386i SYSTEM DESCRIPTION

For Sun386i system, the value of *type* is the constant **Sun**, for the console frame buffer.

Sun386i SYSTEM OPTIONS

–n invoke the full screen login program **logintool**(8), and optionally the "New User Accounts" feature. May only be used on a frame buffer. Unless removed from the console entry in */etc/ttytab*, this option is in effect by default.

FILES

/etc/gettytab

SEE ALSO

login(1), **ioctl**(2), **tty**(4), **fbtab**(5), **gettytab**(5), **svdtab**(5), **ttytab**(5), **init**(8), **logintool**(8)

DIAGNOSTICS

ttyxx: **No such device or address.**

ttyxx: **No such file or directory.**

A terminal which is turned on in the **ttys** file cannot be opened, likely because the requisite lines are either not configured into the system, the associated device was not attached during boot-time system configuration, or the special file in **/dev** does not exist.

NAME

gpconfig – initialize the Graphics Processor

SYOPNSIS

/usr/etc/gpconfig *gpunit* [[**–b**] [**–f**] *fbunit*... [**–u** *microcode-file*]]

DESCRIPTION

gpconfig binds **cgtwo** frame buffers to the GP, (Graphics Processor) and loads and starts the appropriate microcode in the GP. For example, the command line:

/usr/etc/gpconfig gpone0 cgtwo0 cgtwo1

will bind the frame buffer boards **cgtwo0** and **cgtwo1** to the Graphics Processor **gpone0**. The devices **/dev/gpone0a** and **/dev/gpone0b** will then refer to the combination of **gpone** and **cgtwo0** or **cgtwo1** respectively.

The same **cgtwo** frame buffer cannot be bound to more than one GP.

All **cgtwo** frame buffer boards bound to a GP must be configured to the same width and height.

The standard version of the file **/etc/rc.local** contains the following **gpconfig** command line:

/usr/etc/gpconfig gpone0 –f –b cgtwo0

This binds **gpone0** and **cgtwo0** as **gpone0a**, causes **gpone0a** to use the Graphics Buffer Board if it is present, and redirects **/dev/fb** to be **/dev/gpone0a**. If another configuration is desired, edit the command line in **/etc/rc.local** to do the appropriate thing.

It is inadvisable to run the **gpconfig** command while the GP is being used. Unpredictable results may occur. If it is necessary to change the frame buffer bindings to the GP (or to stop using the GP altogether), bring the system down gently, boot single user, edit the **gpconfig** line in the **/etc/rc.local** file, and bring the system back up multiuser.

OPTIONS

–b Configure the GP to use the Graphics Buffer as well. Currently only one GP-to-frame-buffer binding is allowed to use the graphics buffer at a time. Only the last **–b** option in the command line takes effect.

–f Redirect **/dev/fb** to the device formed by binding *gpunit* with **fbunit**. Only the last **–f** option in the command line takes effect.

–u *microcode-file*
Load the specified microcode file instead of the default file from **/usr/lib**.

FILES

/dev/cgtwo[0-9]
/dev/fb
/dev/gpone[0-3][abcd]
/usr/lib/gp1cg2.1024.ucode
/usr/lib/gp1cg2.1152.ucode
/etc/rc.local

SEE ALSO

cgtwo(4S), **gpone**(4S)

NAME

grpck – check group database entries

SYNOPSIS

/usr/etc/grpck [*filename*]

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

grpck checks that a file in **group**(5) does not contain any errors; it checks the **/etc/group** file by default.

FILES

/etc/group

DIAGNOSTICS

Too many/few fields
An entry in the group file does not have the proper number of fields.

No group name
The group name field of an entry is empty.

Bad character(s) in group name
The group name in an entry contains characters other than lower-case letters and digits.

Invalid GID
The group ID field in an entry is not numeric or is greater than 65535.

Null login name
A login name in the list of login names in an entry is null.

Login name not found in password file
A login name in the list of login names in an entry is not in the password file.

First char in group name not lower case alpha
The group name in an entry does not begin witha lower-case letter.

Group name too long
The group name in an entry has more than 8 characters.

SEE ALSO

groups(1), **group**(5), **passwd**(5)

NAME

gxtest – stand alone test for the Sun video graphics board

SYNOPSIS

b /stand/gxtest

DESCRIPTION

gxtest runs stand alone, not under control of the operating system. With the PROM resident monitor in control of the system, type the command:

> b /stand/gxtest

and the monitor boots the video test program into memory. **gxtest** is completely self-explanatory and runs under its own steam. It reports any errors it finds on the screen.

NAME

halt – stop the processor

SYNOPSIS

/usr/etc/halt [**–nqy**]

DESCRIPTION

halt writes out any information pending to the disks and then stops the processor.

halt normally logs the system shutdown to the system log daemon, **syslogd**(8), and places a shutdown record in the login accounting file **/var/adm/wtmp**. These actions are inhibited if the **–n** or **–q** options are present.

OPTIONS

–n Prevent the *sync* before stopping.

–q Do a quick halt. No graceful shutdown is attempted.

–y Halt the system, even from a dialup terminal.

FILES

/var/adm/wtmp login accounting file

SEE ALSO

reboot(8), **shutdown**(8), **syslogd**(8)

NAME

hostrfs – convert IP addresses to RFS format

SYNOPSIS

hostrfs *hostname* [*portnum*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

hostrfs converts IP addresses to a format suitable for use by Remote File Sharing (RFS). It takes a hostname and an optional portnumber and produces an address in the following format:

\x<AF-INET><*portnum*>**<***IP-address***>0000000000000000**

Each field given above is a hex ASCII representation. The **AF_INET** field is the address family which always has the value **0002**. *portnum* is the two-byte TCP port number; if not specified on the command line it defaults to **1450**. *IP-address* is the IP address of the *hostname* given on the command line followed by 16 trailing zeroes.

The output of this command may be directly used as the network address field for the address of an RFS name server in the **rfmaster**(5) file. It may also be used as input to the **nlsadmin** (8) command to initialize the addresses on which the **listener** program listens for service requests.

EXAMPLES

The output of

example% hostrfs wopr

is

\0002145081903509000000000000000000

The output of the command can be used to initialize the network address on which the RFS **listener** program listens for remote service requests, for example:

example# nlsadmin –l 'hostrfs wopr' tcp

SEE ALSO

rfmaster(5), **nlsadmin**(8)

System and Network Administration

NAME

htable – convert DoD Internet format host table

SYNOPSIS

/usr/etc/htable *filename*

DESCRIPTION

htable converts a host table in the format specified by RFC 952 to the format used by the network library routines. Three files are created as a result of running **htable**: **hosts**, **networks**, and **gateways**. The **hosts** file is used by the **gethostent**(3N) routines in mapping host names to addresses. The **networks** file is used by the **getnetent**(3N) routines in mapping network names to numbers. The **gateways** file is used by the routing daemon in identifying "passive" Internet gateways; see **routed**(8C) for an explanation.

If any of the files **localhosts**, **localnetworks**, or **localgateways** are present in the current directory, the file's contents is prepended to the output file without interpretation. This allows sites to maintain local aliases and entries which are not normally present in the master database.

htable is best used in conjunction with the **gettable**(8C) program which retrieves the DoD Internet host table from a host.

FILES

localhosts
localnetworks
localgateways

SEE ALSO

intro(3), **gethostent**(3N), **getnetent**(3N), **gettable**(8C), **routed**(8C)

Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, *DoD Internet Host Table Specification*, RFC 952, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

BUGS

Does not properly calculate the **gateways** file.

NAME

icheck – file system storage consistency check

SYNOPSIS

/usr/etc/icheck [**–s**] [**–b** *numbers*] [*filesystem*]

DESCRIPTION

Note: **icheck** has been superseded for normal consistency checking by **fsck**(8).

icheck examines a file system, builds a bit map of used blocks, and compares this bit map against the free list maintained on the file system. The normal output of **icheck** includes a report of

The total number of files and the numbers of regular, directory, block special and character special files.

The total number of blocks in use and the numbers of single-, double-, and triple-indirect blocks and directory blocks.

The number of free blocks.

The number of blocks missing; that is, not in any file nor in the free list.

With the **–s** option **icheck** ignores the actual free list and reconstructs a new one by rewriting the superblock of the file system. The file system should be dismounted while this is done; if this is not possible (for example if the root file system has to be salvaged) care should be taken that the system is quiescent and that it is rebooted immediately afterwards so that the old, bad in-core copy of the superblock will not continue to be used. Notice also that the words in the superblock which indicate the size of the free list and of the i-list are believed. If the superblock has been curdled these words will have to be patched. The **–s** option suppresses the normal output reports.

Following the **–b** option is a list of block numbers; whenever any of the named blocks turns up in a file, a diagnostic is produced.

icheck is faster if the raw version of the special file is used, since it reads the i-list many blocks at a time.

SEE ALSO

fs(5), **clri**(8), **dcheck**(8), **fsck**(8), **ncheck**(8)

DIAGNOSTICS

For duplicate blocks and bad blocks (which lie outside the file system) **icheck** announces the difficulty, the i-number, and the kind of block involved. If a read error is encountered, the block number of the bad block is printed and **icheck** considers it to contain 0.

Bad freeblock
means that a block number outside the available space was encountered in the free list.

***n* dups in free**
means that *n* blocks were found in the free list which duplicate blocks either in some file or in the earlier part of the free list.

BUGS

Since **icheck** is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

It believes even preposterous superblocks and consequently can get core images.

The system should be fixed so that the reboot after fixing the root file system is not necessary.

NAME

idload – RFS user and group mapping

SYNOPSIS

idload [**–n**] [**–g** *g_rules*] [**–u** *u_rules*] [*directory*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

idload is used on Remote File Sharing (RFS) servers to build translation tables for user and group IDs. It takes your **/etc/passwd** and **/etc/group** files and produces translation tables for user and group IDs from remote machines, according to the rules set down in the *u_rules* and *g_rules* files. If you are mapping by user and group name, you will need copies of remote **/etc/passwd** and **/etc/group** files. If no rules files are specified, remote user and group IDs are mapped to MAXUID+1. This is an ID number that is one higher than the highest number you could assign on your system.

By default, the remote password and group files are assumed to reside in **/usr/nserve/auth.info/***domain/host***/[passwd | group].** The *directory* argument indicates that some directory structure other than **/usr/nserve/auth.info** contains the *domain/host* **passwd** and **group** files. *host* is the name of the host the files are from and *domain* is the domain where *host* can be found.

This command is restricted to the super-user.

This command is run automatically when the first remote mount is done of a remote resource (see **mount**(8)).

If any of the following are true, an error message will be sent to standard error.

- Neither rules files can be found or opened.
- There are syntax errors in the rules file.
- There are semantic errors in the rules file.
- Host information could not be found.
- The command is not run with super-user privileges.

Partial failures will display a warning message, although the process will continue.

OPTIONS

–n Do not produce a translation table, however, send a display of the ID mapping to the standard out. This is used to do a trial run of the mapping.

–u *u_rules* The *u_rules* file contains the rules for user ID translation. The default rules file is **/usr/nserve/auth.info/uid.rules.**

–g *g_rules* The *g_rules* file contains the rules for group ID translation. The default rules file is **/usr/nserve/auth.info/gid.rules.**

USAGE

Rules

The rules files have two types of sections, both optional: **global** and **host**. There can be only one global section, though there can be one host section for each host you want to map.

The **global** section describes the default conditions for translation for any machines that are not explicitly referenced in a **host** section. If the global section is missing, the default action is to map all remote user and group IDs from undefined hosts to MAXUID+1. The syntax of the first line of the **global** section is:

global

A **host** section is used for each client machine or group of machines that you want to map differently from the global definitions. The syntax of the first line of each **host** section is:

host*name*[...]

where *name* is replaced by the full name(s) of a host (*domain.hostname*).

The format of a rules file is described below. All lines are optional, but must appear in the order shown.

global
default *local* | **transparent**
exclude
[*remote_id–remote_id*] | [*remote_id*]
map [*remote_id*:*local*]

host *domain.hostname* [*domain.hostname*...]
default *local* | **transparent**
exclude [*remote_id–remote_id*] | [*remote_id*] | [*remote_name*]
map [*remote*:*local*] | *remote* | **all**

Each of these instruction types is described below.

The line

default *local* | **transparent**

defines the mode of mapping for remote users that are not specifically mapped in instructions in other lines. **transparent** means that all remote user and group IDs will have the same numeric value locally unless they appear in the **exclude** instruction. *local* can be replaced by a local user name or ID to map all users into a particular local name or ID number. If the default line is omitted, all users that are not specifically mapped are mapped into a "special guest" login ID .

The line

exclude [*remote_id–remote_id*] | [*remote_id*] | [*remote_name*]

defines remote IDs that will be excluded from the **default** mapping. The **exclude** instruction must precede any **map** instructions in a block. You can use a range of ID numbers, a single ID number, or a single name. (*remote_name* cannot be used in a global block.)

The line

map [*remote*:*local*] | *remote* | **all**

defines the local IDs and names that remote IDs and names will be mapped into. *remote* is either a remote ID number or remote name; *local* is either a local ID number or local name. Placing a colon between a *remote* and a *local* will give the value on the left the permissions of the value on the right. A single *remote* name or ID will assign the user or group permissions of the same local name or ID. **all** is a predefined alias for the set of all user and group IDs found in the local **/etc/passwd** and **/etc/group** files. You cannot map by remote name in **global** blocks.

Note: **idload** will always output warning messages for '**map all**', since password files always contain multiple administrative user names with the same ID number. The first mapping attempt on the ID number will succeed, all subsequent attempts will fail.

RFS does not need to be running to use **idload**.

EXIT STATUS

On successful completion, **idload** will produce one or more translation tables and return a successful exit status. If **idload** fails, the command will return an unsuccessful exit status without producing a translation table.

FILES

/etc/passwd
/etc/group
/usr/nserve/auth.info/*domain***/***host***/[user | group]**
/usr/nserve/auth.info/vid.rules
/usr/nserve/auth.info/gid.rules

SEE ALSO

mount(8)

NAME

ifconfig – configure network interface parameters

SYNOPSIS

/usr/etc/ifconfig *interface* [*address_family*] [*address* [*dest_address*]] [**netmask** *mask*]
[**broadcast** *address*] [**up**] [**down**] [**trailers**] [**–trailers**] [**arp**] [**–arp**] [**private**]
[**–private**] [**metric** *n*]

/usr/etc/ifconfig *interface* [*protocol_family*]

DESCRIPTION

ifconfig is used to assign an address to a network interface and/or to configure network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters. Used without options, **ifconfig** displays the current configuration for a network interface. If a protocol family is specified, **ifconfig** will report only the details specific to that protocol family. Only the super-user may modify the configuration of a network interface.

The *interface* parameter is a string of the form *nameunit*, for example **ie0**. The interface name ''*–a*'' is reserved, and causes the remainder of the arguments to be applied to each address of each interface in turn.

Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, the parameters and addresses are interpreted according to the rules of some address family, specified by the *address_family* parameter. The address families currently supported are **ether** and **inet**. If no address family is specified, **inet** is assumed.

For the TCP/IP family (**inet**), the address is either a host name present in the host name data base (see **hosts**(5)) or in the Network Interface Service (NIS) map **hosts**, or a TCP/IP address expressed in the Internet standard "dot notation". Typically, an Internet address specified in dot notation will consist of your system's network number and the machine's unique host number. A typical Internet address is **192.9.200.44**, where **192.9.200** is the network number and **44** is the machine's host number.

For the **ether** address family, the address is an Ethernet address represented as *x*:*x*:*x*:*x*:*x*:*x* where *x* is a hexadecimal number between 0 and ff. Only the super-user may use the **ether** address family.

If the *dest_address* parameter is supplied in addition to the *address* parameter, it specifies the address of the correspondent on the other end of a point to point link.

OPTIONS

up Mark an interface "up". This happens automatically when setting the first address on an interface. The **up** option enables an interface after an **ifconfig down**, reinitializing the hardware.

down Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

trailers This flag used to cause a non-standard encapsulation of inet packets on certain link levels. Sun drivers no longer use this flag, but it is ignored for compatibility.

–trailers Disable the use of a "trailer" link level encapsulation.

arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between TCP/IP addresses and 10Mb/s Ethernet addresses.

–arp Disable the use of the Address Resolution Protocol.

private Tells the **in.routed** routing daemon (see **routed**(8C)) that the interface should not be advertised.

–private Specify unadvertised interfaces.

metric *n* Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (**routed**(8C)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

netmask mask (**inet** only) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation address, or with a pseudo-network name listed in the network table **networks**(5). The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion. If a '+' (plus sign) is given for the netmask value, then the network number is looked up in the NIS **netmasks.byaddr** map (or in the **/etc/netmasks**) file if not running the NIS service.

broadcast *address*

(**inet** only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 0's. A + (plus sign) given for the broadcast value causes the broadcast address to be reset to a default appropriate for the (possibly new) address and netmask. Note that the arguments of **ifconfig** are interpreted left to right, and therefore

ifconfig –a netmask + broadcast +

and

ifconfig –a broadcast + netmask +

may result in different values being assigned for the interfaces' broadcast addresses.

EXAMPLES

If your workstation is not attached to an Ethernet, the **ie0** interface should be marked "down" as follows:

ifconfig ie0 down

To print out the addressing information for each interface, use

ifconfig –a

To reset each interface's broadcast address after the netmasks have been correctly set, use

ifconfig –a broadcast +

FILES

/dev/nit
/etc/netmasks

SEE ALSO

intro(3), **ethers**(3N), **arp**(4P), **hosts**(5), **netmasks**(5), **networks**(5) **netstat**(8C), **rc**(8), **routed**(8C)

DIAGNOSTICS

Messages indicating the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

imemtest – stand alone memory test

SYNOPSIS

b /stand/imemtest

DESCRIPTION

imemtest runs stand alone, not under control of the operating system. With the PROM resident monitor in control of the system, type the command:

> b /stand/imemtest

and the monitor boots the memory test program into memory. **imemtest** is completely self-explanatory. It prompts for all start and end addresses, and after that it runs under its own steam. It reports any errors it finds on the screen.

NAME

inetd – Internet services daemon

SYNOPSIS

/usr/etc/inetd [**–d**] [*configuration-file*]

DESCRIPTION

inetd, the Internet services daemon, is normally run at boot time by the **/etc/rc.local** script. When started **inetd** reads its configuration information from *configuration-file*, the default being **/etc/inetd.conf**. See **inetd.conf**(5) for more information on the format of this file. It listens for connections on the Internet addresses of the services that its configuration file specifies. When a connection is found, it invokes the server daemon specified by that configuration file for the service requested. Once a server is finished, **inetd** continues to listen on the socket (except in some cases which will be described below).

Depending on the value of the "wait-status" field in the configuration line for the service, **inetd** will either wait for the server to complete before continuing to listen on the socket, or immediately continue to listen on the socket. If the server is a "single-threaded" datagram server (a "wait-status" field of "wait"), **inetd** must wait. That server will handle all datagrams on the socket. All other servers (stream and ×lti-threaded" data-gram, a "wait-status" field of "nowait") operate on separate sockets from the connection request socket, thus freeing the listening socket for new connection requests.

Rather than having several daemon processes with sparsely distributed requests each running concurrently, **inetd** reduces the load on the system by invoking Internet servers only as they are needed.

inetd itself provides a number of simple TCP-based services. These include **echo**, **discard**, **chargen** (character generator), **daytime** (human readable time), and **time** (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). For details of these services, consult the appropriate RFC, as listed below, from the Network Information Center.

inetd rereads its configuration file whenever it receives a hangup signal, **SIGHUP**. New services can be activated, and existing services deleted or modified in between whenever the file is reread.

SEE ALSO

inetd.conf(5), **comsat**(8C), **ftpd**(8C), **rexecd**(8C), **rlogind**(8C), **rshd**(8C), **telnetd**(8C), **tftpd**(8C)

Postel, Jon, *Echo Protocol*, RFC 862, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Postel, Jon, *Discard Protocol*, RFC 863, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Postel, Jon, *Character Generator Protocol*, RFC 864, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Postel, Jon, *Daytime Protocol*, RFC 867, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Postel, Jon, and Ken Harrenstien, *Time Protocol*, RFC 868, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

NAME

infocmp – compare or print out terminfo descriptions

SYNOPSIS

infocmp [**–cdnILCruvV1**] [**–sd**] [**–si**] [**–sl**] [**–sc**] [**–w** *width*] [**–A** *directory*] [**–B** *directory*] [*termname* ...]

SYNOPSIS

/usr/5bin/infocmp *arguments*

Note: *arguments* to **/usr/5bin/infocmp** are the same as those for **infocmp**, above.

AVAILABILITY

The System V version of this command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

infocmp compares a binary **terminfo**(5V) entry with other terminfo entries, rewrites a **terminfo** description to take advantage of the **use=***field*, or prints out a **terminfo** description from the corresponding binary file in a variety of formats. It displays boolean fields first, then numeric fields, then string fields.

It can also convert a **terminfo** entry to a **termcap**(5) entry; the **–C** flag causes **infocmp** to perform this conversion. Some **termcap** variables are not supported by **terminfo**, but those that can be derived from *terminfo* variables are displayed. Not all **terminfo** capabilities are translated either; only those that are allowed in a **termcap** entry are normally displayed. Specifying the **–r** option eliminates this restriction, allowing all capabilities to be displayed in **termcap** form.

Because padding is collected at the beginning of a capability, not all capabilities are displayed. Since mandatory padding is not supported by **terminfo** and **termcap** strings are not as flexible, it is not always possible to convert a **terminfo** string capability into an equivalent working **termcap** capability. Also, a subsequent conversion of the **termcap** file back into **terminfo** format will not necessarily reproduce the original source; **infocmp** attempts to convert parameterized strings, and comments out those that it can not.

Some common **terminfo** parameter sequences, their **termcap** equivalents, and some terminal types which commonly have such sequences, are:

Terminfo	Termcap	Representative Terminals
%p1%c	%.	adm
%p1%d	%d	hp, ANSI standard, vt100
%p1%'x'%+%c	%+x	concept
%i	%i	ANSI standard, vt100
%p1%?%'x'%>%t%p1%'y'%+%;	%>xy	concept
%p2 is printed before %p1	%r	hp

If no *termname* arguments are given, the environment variable **TERM** is used for all expected *termname* arguments.

OPTIONS

Default Options

If no options are specified and either zero or one *termname* is specified, the **–I** option is assumed to be in effect. If more than one *termname* is specified, the **–d** option is assumed.

Comparison Options

infocmp compares the description of the first terminal *termname* with each of the descriptions for terminals listed in subsequent *termname* arguments. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: **F** for boolean variables, **–1** for integer variables, and **NULL** for string variables.

–c Produce a list of capabilities common to both entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the **–u** option is worth using.

-d Produce a list of capabilities that differ between descriptions.

-n Produce a list of capabilities in neither entry.

Source Listing Options

The **-I**, **-L**, and **-C** options produce a source listing for each terminal named.

-I Use the **terminfo** names.

-L Use the long C variable name listed in **<term.h>**.

-C Display only those capabilities that have **termcap** equivalents, using the **termcap** names and displaying them in **termcap** form whenever possible.

The source produced by the **-C** option may be used directly as a **termcap** entry, but not all of the parameterized strings may be changed to the **termcap** format. All padding information for strings is collected together and placed at the beginning of the string where **termcap** expects it. Mandatory padding (padding information with a trailing '/') will become optional.

-r When using **-C**, display all capabilities, not just those capabilities that have **termcap** equivalents.

-u Produce a **terminfo** source description for the first named terminal which is relative to the descriptions given by the entries for all terminals named subsequently on the command line, by analyzing the differences between them, and producing a description with **use=** fields for the other terminals. In this manner, it is possible to retrofit generic terminfo entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times or by different people so that each description is a full description, using **infocmp** will show what can be done to change one description to be relative to the other.

A capability is displayed with an at-sign (**@**) if it no longer exists in the first terminal, but one of the other terminal entries contains a value for it. A capability's value gets printed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries has a different value for that capability.

The order of the other *termname* entries is significant. Since the terminfo compiler **tic**(8V) does a left-to-right scan of the capabilities, specifying two **use=** entries that contain differing entries for the same capabilities will produce different results, depending on the order in which they are given. **infocmp** flags any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability after a **use=** entry that contains it, will cause the second specification to be ignored. Using **infocmp** to recreate a description can be a useful check to make sure that everything was specified correctly in the original.

Specifying superfluous **use=** slows down the comparison, but is not fatal; **infocmp** flags superfluous **use=** fields.

Sorting Options

-sd Sort fields in the order that they are stored in the **terminfo** database.

-si Sort fields by **terminfo** name.

-sl Sort fields by the long C variable name.

-sc Sort fields by the **termcap** name.

If no sorting option is given, fields are sorted alphabetically by the **terminfo** name within each type, except in the case of the **-C** or the **-L** options, which cause the sorting to be done by the **termcap** name or the long C variable name, respectively.

Changing Databases

The location of the compiled **terminfo** database is taken from the environment variable TERMINFO. If the variable is not defined, or if the terminal is not found in that location, the system **terminfo** database, usually in **/usr/share/lib/terminfo**, is used. The options **–A** and **–B** may be used to override this location. With these options, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people.

–A Set TERMINFO for the first *termname* argument.

–B Set TERMINFO for the remaining *termname* arguments.

Other Options

–v Print out tracing information on the standard error.

–V Print out the version of the program in use on the standard error and exit.

–1 Print fields out one to a line. Otherwise, fields are printed several to a line to a maximum width of 60 characters.

–w *width*

Change the output to *width* characters.

FILES

/usr/share/lib/terminfo/?/*

compiled terminal description database

/usr/5include/term.h

SEE ALSO

curses(3V), **termcap**(5), **terminfo**(5V), **tic**(8V)

DIAGNOSTICS

malloc is out of space!

There was not enough memory available to process all the terminal descriptions requested. Run **infocmp** in several smaller stages (with fewer *termname* arguments).

use= order dependency found:

A value specified in one relative terminal specification was different from that in another relative terminal specification.

'use=*term***' did not add anything to the description.**

A relative terminal name did not contribute anything to the final description.

must have at least two terminal names for a comparision to be done.

The **–u**, **–d** and **–c** options require at least two terminal names.

NAME

init – process control initialization

SYNOPSIS

/usr/etc/init [**–bs**]

DESCRIPTION

init is invoked inside the operating system as the last step in the boot procedure. It normally runs the sequence of commands in the script **/etc/rc.boot** (see **rc**(8)) to check the file system. If passed the **–b** option from the boot program, **init** skips this step. If the file system check succeeds or is skipped, **init** runs the commands in **/etc/rc** and **/etc/rc.local** to begin multiuser operation; otherwise it commences single-user operation by giving the super-user a shell on the console. It is possible to pass the **–s** parameter from the boot program to **init** so that single-user operation is commenced immediately.

Whenever a single-user shell is created, and the system is running as a secure system, the **init** program demands the super-user password. This is to prevent an ordinary user from invoking a single-user shell and thereby circumventing the system's security. Logging out (for instance, by entering an EOT) causes **init** to proceed with a multi-user boot. The super-user password is demanded whenever the system is running secure as determined by **issecure**(3), or the console terminal is not labeled "secure" in **/etc/ttytab**.

Whenever single-user operation is terminated (for instance by killing the single-user shell) **init** runs the scripts mentioned above.

In multi-user operation, **init**'s role is to create a process for each terminal port on which a user may log in. To begin such operations, it reads the file **/etc/ttytab** and executes a command for each terminal specified in the file. This command will usually be **/usr/etc/getty**. **getty**(8) opens and initializes the terminal line, reads the user's name and invokes **login**(1) to log in the user and execute the shell.

Ultimately the shell will terminate because it received EOF, either explicitly, as a result of hanging up, or from the user logging out. The main path of **init**, which has been waiting for such an event, wakes up and removes the appropriate entry from the file **/etc/utmp**, which records current users. **init** then makes an entry in **/var/adm/wtmp**, which maintains a history of logins and logouts. The **/var/adm/wtmp** entry is made only if a user logged in successfully on the line. Then the appropriate terminal is reopened and the command for that terminal is reinvoked.

init catches the *hangup* signal (SIGHUP) and interprets it to mean that the file **/etc/ttytab** should be read again. The shell process on each line which used to be active in **/etc/ttytab** but is no longer there is terminated; a new process is created for each added line; lines unchanged in the file are undisturbed. Thus it is possible to drop or add terminal lines without rebooting the system by changing **/etc/ttytab** and sending a *hangup* signal to the **init** process: use '**kill –HUP 1**'.

init terminates multi-user operations and resumes single-user mode if sent a terminate (SIGTERM) signal: use '**kill –TERM 1**'. If there are processes outstanding which are deadlocked (due to hardware or software failure), **init** does not wait for them all to die (which might take forever), but times out after 30 seconds and prints a warning message.

init ceases to create new processes, and allows the system to slowly die away, when sent a terminal stop (SIGTSTP) signal: use '**kill –TSTP 1**'. A later hangup will resume full multi-user operations, or a terminate will initiate a single-user shell. This hook is used by **reboot**(8) and **halt**(8).

Whenever it reads **/etc/ttytab**, **init** will normally write out an old-style **/etc/ttys** file reflecting the contents of **/etc/ttytab**. This is required in order that programs built on earlier versions of SunOS that read the **/etc/ttys** file (for example, programs using the **ttyslot**(3V) routine, such as **shelltool (1)**) may continue to run. If it is not required that such programs run, **/etc/ttys** may be made a link (hard or symbolic) to **/etc/ttytab** and **init** will not write to **/etc/ttys**.

init's role is so critical that if it dies, the system will reboot itself automatically. If, at bootstrap time, the **init** program cannot be located, the system will print an error message and panic.

FILES

/dev/console
/dev/tty*
/etc/utmp
/var/adm/wtmp
/etc/ttytab
/etc/rc
/etc/rc.local
/etc/rc.boot
/usr/etc/getty

SEE ALSO

kill(1), **login**(1), **sh**(1), **shelltool**(1), **issecure**(3), **ttyslot**(3V), **ttytab**(5), **getty**(8), **halt**(8), **rc**(8), **reboot**(8), **shutdown**(8)

DIAGNOSTICS

command **failing, sleeping.**

A process being started to service a line is exiting quickly each time it is started. This is often caused by a ringing or noisy terminal line. **init** will sleep for 30 seconds, then continue trying to start the process.

WARNING: Something is hung (won't die); ps axl advised.

A process is hung and could not be killed when the system was shutting down. This is usually caused by a process which is stuck in a device driver due to a persistent device error condition.

NAME

installboot – install bootblocks in a disk partition

SYNOPSIS

/usr/mdec/installboot [**–lvt**] *bootfile protobootblk bootdevice*

DESCRIPTION

The **boot**(8S) program is loaded from disk by bootblock code which resides in the bootblock area of a disk partition. In order for the bootblock code to read the boot program (usually **/boot**) it is necessary for it to know the block numbers occupied by the boot program. Previous versions of the bootblock code could find **/boot** by interpreting the file system on the partition from which it was being booted, but this is no longer so.

installboot plugs the block numbers of the boot program into a table in the bootblock code, and writes the modified bootblock code onto the disk. Note: **installboot** must be run every time the boot program is reinstalled, since in general, the block list of the boot program will change each time it is written.

bootfile is the name of the boot program, usually **/boot**. *protobootblk* is the name of the bootblock code into which the block numbers of the boot program are to be inserted. The file read in must have an **a.out**(5) header, but it will be written out to the device with the header removed. *bootdevice* is the name of the disk device onto which the bootblock code is to be installed.

OPTIONS

–l Print out the list of block numbers of the boot program.

–t Test. Display various internal test messages.

–v Verbose. Display detailed information about the size of the boot program, etc.

EXAMPLE

To install the bootblocks onto the root partition on a Xylogics disk:

```
example% cd /usr/mdec
example% installboot –vlt /boot bootxy /dev/rxy0a
```

For an SD disk, you would use **bootsd** and **/dev/rsd0a**, respectively, in place of **bootxy** and **/dev/rxy0a**.

SEE ALSO

od(1V), **a.out**(5) **boot**(8S), **bootparamd**(8), **init**(8), **kadb**(8S), **monitor**(8S), **ndbootd**(8C), **rc**(8), **reboot**(8)

System and Network Administration

Installing SunOS 4.1

NAME

install_small_kernel – install a small, pre-configured kernel

SYNOPSIS

/usr/etc/install/install_small_kernel [*hostname*] ...

DESCRIPTION

install_small_kernel is a script that installs a small, pre-configured kernel, **GENERIC_SMALL** on a host. This kernel supports approximately four users, and is only available for the following configurations:

Sun-3/50 and Sun-3/60 systems with up to 2 SCSI disks, 1 SCSI tape
Sun-3/80 systems with up to 4 SCSI disks, 1 SCSI tape
Sun-4/110 systems with up to 2 SCSI disks, 1 SCSI tape
SPARCsystem 330 systems with up to 4 SCSI disks, 1 SCSI tape
SPARCstation 1 systems with up to 4 SCSI disks. 1 floppy drive and 2 SCSI tapes

If *hostname* is a server that does not fit any of the above configurations, **install_small_kernel** can be used to install the small kernel on its clients.

If no hostnames are specified, **install_small_kernel** cycles through all the clients configured for a server to determine the small kernel installs to be made. If the '**small_kernel**' flag in the client file, **/etc/install/client.***hostname* is set to '**yes**', that client will not be processed. To force re-installation of a small kernel on any clients, simply call **install_small_kernel** with the appropriate client names.

install_small_kernel prompts for confirmation before actually doing the install on any host.

install_small_kernel is executable from the miniroot, as well as single-user and multi-user modes. It supports standalone and server configuration in all cases, but dataless systems are supported in multi-user mode only. This script is restricted to the super-user.

FILES

/usr/sys/sun*arch***/conf/GENERIC_SMALL**
kernel configuration file for *arch* **/usr/install/client.***hostname*

SEE ALSO

add_client(8), **add_services**(8), **rm_client**(8), **suninstall**(8)

System and Network Administration

NAME

installtxt, gencat – create a message archive

SYNOPSIS

/usr/etc/installtxt [[–]**d**|**c**|**r**|**t**|**x**|**i** [**ouvs**]]] *message-archive*... [*source-message-file*]

/usr/etc/gencat *catfile msgfile*...

DESCRIPTION

installtxt converts each *source-message-file* into a binary format message archive. At the same time, if necessary, **installtxt** maintains groups of files (member files) combined into a single message archive. **installtxt** is normally used to create and update message archives used by the run-time message handling facility **gettext**(3).

gencat performs the same function as **installtxt**, but supports the X/Open catalog source format.

installtxt creates the message archive in *message-archive*. If the message archive does not exist, it is created by the **–c** option. *source-message-file* contains source versions of the target strings. On successful completion of an update operation of **installtxt**, the message archive will have been updated with details of the formatted version of each *source-message-file*. If *message-archive* does not contain the full pathname of the run-time location of the message catalog, it will have to be moved to the appropriate locale directory before applications using the archive are activated.

gencat merges the message text source files (*msgfile*...) into a formatted message catalog *catfile*. *catfile* is created if it does not already exist. If *catfile* does exist, its messages are included in the new *catfile*. If set and message numbers collide, the new message-text defined in *msgfile* will replace the old message text currently contained in *catfile*. The output formats of both *message_archive* and *catfile* are the same. However it should be noted that on a per-application basis, it is not intended that the output forms of these two utilities should be mixed, and the consequence of doing so is undefined.

OPTIONS

The following options and modifiers apply to **installtxt** only. For **installtxt** you must indicate only one of: **c**, **d**, **r**, **t**, or, **x**, which may be followed by one or more **Modifiers**, **o**, **u**, or , **v**.

The options are:

c Create. The member file called *source-message-file* is being made for the first time in the message archive. It should not exist already.

d Delete the named member files from *message archive*. Note that individual messages can be deleted by entering an empty value after the message-id selecting the message to be deleted. With the **v** option these deletions are notified on the standard output.

r Replace the named member files in the message archive. This allows the existing *message archive* to be merged with new versions of messages. No new message will be added to the message archive unless each message-tag in the *source-message-file* is unique in the active domain. If the member file contains a message-tag that is not unique within the active domain, **installtxt** will fail and the contents of the active message archive will not be altered.

t Table of contents. Produces a list on the standard output of all member files in *message_archive*.

x Extract. If no names are given, all member files in the message archive are extracted into the current directory; if names are given, only those files are extracted. In neither case does **x** alter the message archive. The extracted member files will be returned in their original source format. It is possible for the **–x** option to lose comments that were contained in the original source message file. In addition, overlong lines may be escaped (using \n) at a point that is different from the original source, although the end result will logically be the same string.

Modifiers

o Old date. When member files are extracted with the **x** option, set the "last modified" date to the date recorded in the message archive.

u Update. Replace only those member files that have changed since they were put in the message archive. Used with the **r** option.

v Verbose. When used with the **c**, **r**, or **d** option, give a file-by-file description of the creation of a new *message archive* file from the old version and the constituent member files. When used with **x**, give a file-by-file description of the extraction of message archive member files. When used with **t**, print information about the size and creation date of the message archive, as well as a count of the number of target strings in the message-archive.

USAGE

source-message-file consists of one or more lines of text, with each line containing either a comment, a directive or a text line. The format of a comment line is:

"$ %s", *comment*

A line beginning with a dollar sign ($), followed by a *blank* character streated as a comment line. The format of directives is:

"$%s %s", *control-type*, *value*

Directives should be directly preceded by a dollar sign ($), and followed by an optional value. There is one *blank* character between the directive and its value. The following directives are recognized:

$separator *c*

This directive specifies an optional separator character that will subsequently be used in the following text lines to separate the message identifier from the target string. There is one *blank* character between **separator** and the separator character itself. If this line is absent then the default separator is the *blank* character. Only the first occurrence of this character on one text line will be interpreted, for example:

$separator :
12345:Bonjour: Mon ami

would declare the message identifier to be 12345, the target string would contain the second ":".

$domain *domain*

This directive states that all following target strings are contained within a domain of the object message file as described by *domain*. *domain* can be any string of up to {PATH_MAX} bytes in length.

$quote *c* This directive specifies an optional quote character *c*, which can be used to surround both *message_string* and *message_identifier* . By default, or if an empty **$quote** directive is supplied, no quoting of *message_string* will be recognized. If the **$quote** directive is given then all message strings must contain pairs of quotes, although quotes around the message_identifier are still optional after the directive.

The format of the text line is:

"%s%s%s", *message_identifier*, *separator_character*, *message_string*

Each line defines a message identifier and a target string pair.

Empty lines in a source text file are ignored. If a *message_identifier* starts with a dollar ($) character, then that dollar character must be escaped with a backslash (\$). Any other form of input line syntax is illegal and will cause **installtxt** to exit with the error value.

Message strings and message identifiers can contain the special characters and escape sequences as defined in the following table:

Description	Symbol
newline	**\n**
tab	**\t**
vertical-tab	**\v**
backspace	**\b**
carriage-return	**\r**
form-feed	**\f**
backslash	****
bit pattern	**\ddd**

The escape sequence *\ddd* consists of backslash followed by 1, 2 or 3 octal digits, which are used to specify the value of the desired character. If *message_identifier* contains the separator character then it must be escaped with a backslash (\) character. If the character following a backslash is not one of those specified, the effect is unspecified.

Backslash, \, followed by a NEWLINE character is used to continue an individual string on the following line. Both *message_identifier* and *message_string* may be continued over lines in this way. *message_string* is stored in *object_file* in an implementation specific way. If *message_string* is empty, and *separator* is present, a null string is stored in *object_file*.

msgfile must be in the X/Open **gencat** format.

EXAMPLES

```
# /bin/sh script
# The following creates a message archive in the file messages.general
installtxt –cv messages.general input
#
```

FILES

/etc/locale/LC_MESSAGES/*locale/domain*
: standard private location for message archive/catalog in locale *locale* and domain *domain*

/usr/share/lib/locale/LC_MESSAGES
: standard shared location for message archive/catalog in locale *locale* and domain *domain*

SEE ALSO

catgets(3), **gettext**(3), **setlocale**(3V), **locale**(5)

X/Open Portability Guide Issue 2

NAME

intr – allow a command to be interruptible

SYNOPSIS

intr [**–anv**] [**–t** *seconds*] *command* [*arguments*]

DESCRIPTION

intr executes *command* after altering the execution environment to make *command* to be interrutable.

Since interactive commands are by default interruptable, **intr** is intended for use as a wrapper around commands started by the **/etc/rc** files; commands spawned from these files are not interruptable by default. It has no other intended use than as a wrapper around **/etc/rc** commands.

The following signals are ignored as a result of wrapping **intr** around a command:

SIGTSTP terminal generated stop signal

SIGTTIN background read

SIGTTOU background write

The following signals are reset to their default actions:

SIGINT interrupt signal

SIGQUIT quit signal

OPTIONS

–v Echo the command in the form ' *command*' (note leading SPACE).

–a Echo the command and its arguments.

–n Do not echo a NEWLINE after the command or arguments (for example '**echo –n ...**').

–t *secs* Arrange to have a **SIGALRM** signal delivered to the command in *secs* seconds.

EXAMPLES

All of these examples assume that they are in an **/etc/rc** file, that is, talking to the console, and not run interactively. The following example runs **fsck**(8) but allow it to be killed from the console:

intr fsck –p –w / /usr

Echoing is provided so that

ypbind; echo –n ' ypbind'

can be replaced with

intr –vn ypbind

Timeouts are provided so that the machine will not hang at boot:

intr –t 10 rdate *date_host*

SEE ALSO

echo(1V), **login**(1), **init**(8), **rc**(8)

BUGS

The **–v** option is a kludge.

NAME

iostat – report I/O statistics

SYNOPSIS

iostat [**–cdDIt**] [**–l** *n*] [*disk* ...] [*interval* [*count*]]

DESCRIPTION

iostat can iteratively report terminal and disk I/O activity, as well as CPU utilization. The first report is for all time since a reboot and each subsequent report is for the prior interval only.

In order to compute this information, the kernel maintains a number of counters. For each disk, seeks and data transfer completions and number of words transferred are counted; for terminals collectively, the number of input and output characters are counted. Also, at each clock tick, the state of each disk is examined and a tally is made if the disk is active. The kernel also provides approximate transfer rates of the devices.

OPTIONS

iostat's activity class options default to **tdc** (terminal, disk, and CPU). If any activity class options are specified, the default is completely overridden. Therefore, if only **–d** is specified, neither terminal nor CPU statistics will be reported. The last disk option specified (either **–d** or **–D**) is the only one that is used.

–c Report the percentage of time the system has spent in user mode, in user mode running low priority processes, see **nice**(1), in system mode, and idling.

–d For each disk, report the number of kilobytes transferred per second, the number of transfers per second, and the milliseconds per average seek (see BUGS below).

–D For each disk, report the reads per second, writes per second, and percentage disk utilization.

–I Report the counts in each interval, rather than reporting rates.

–t Report the number of characters read and written to terminals.

–l *n* Limit the number of disks included in the report to *n*; the disk limit defaults to 4. Note: disks explicitly requested (see *disk* below) are not subject to this disk limit.

disk Explicitly specify the disks to be reported; in addition to any explicit disks, any active disks up to the disk limit (see **–l** above) will also be reported.

interval Report once each *interval* seconds.

count Only print *count* reports.

FILES

/dev/kmem
/vmunix

SEE ALSO

vmstat(8)

BUGS

Milliseconds per average seek is an approximation based on the disk (not the controller) transfer rate. Therefore, the seek time will be over-estimated in systems with slower controllers.

NAME

ipallocd – Ethernet-to-IP address allocator

SYNOPSIS

/usr/etc/rpc.ipallocd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

ipallocd is a daemon that determines or temporarily allocates IP addresses within a network segment. The service is only available on the system which is home to the address authority for the network segment, currently the Network Interface Service (NIS) master of the **hosts.byaddr** map although the service is not tied to the NIS service. It has complete knowledge of the hosts listed in the NIS service, and, if the system is running the name server, of any hosts listed in internet domain tables automatically accessed on that host through the standard library **gethostent**(3N) call.

This protocol uses DES authentication (the Sun Secure RPC protocol) to restrict access to this function. The only clients privileged to allocate addresses are those whose net IDs are in the networks group. For machine IDs, the machine must be an NIS server.

The daemon uses permanent entries in the **/etc/ethers** and **/etc/hosts** files when they exist and are usable. In other cases, such as when a system is new to the network, **ipallocd** enters a temporary mapping in a local cache. Entries in the cache are removed when there have been no references to a given entry in the last hour. This cache survives system crashes so that IP addresses remain consistent.

The daemon also provides corresponding IP address to name mapping.

If the file **/etc/ipalloc.netrange** exists, **ipallocd** refuses to allocate addresses on networks not listed in the **netrange** file, or for which no free address is available.

FILES

/etc/ipalloc.cache	temporary cache
/etc/ipalloc.netrange	optional file to allocate network addresses

SEE ALSO

ipalloc(3R), **pnp**(3R), **ipalloc.netrange**(5), **ipallocd**(8C), **netconfig**(8C), **pnpboot**(8C), **rarpd**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

kadb – adb-like kernel and standalone-program debugger

SYNOPSIS

> b kadb [–d] [*boot-flags* **]**

DESCRIPTION

kadb is an interactive debugger that is similar in operation to **adb**(1), and runs as a standalone program under the PROM monitor. You can use **kadb** to debug the kernel, or to debug any standalone program.

Unlike **adb**, **kadb** runs in the same supervisor virtual address space as the program being debugged — although it maintains a separate context. The debugger runs as a *coprocess* that cannot be killed (no '**:k**') or rerun (no '**:r**'). There is no signal control (no '**:i**', '**:t**', or '**$i**'), although the keyboard facilities (CTRL–C, CTRL–S, and CTRL–Q) are simulated.

While the kernel is running under **kadb**, the abort sequence (**L1-A** or **BREAK**) drops the system into **kadb** for debugging — as will a system panic. When running other standalone programs under **kadb**, the abort sequence will pass control to the PROM monitor. **kadb** is then invoked from the monitor by jumping to the starting address for **kadb** found in **/usr/include/debug/debug.h** The following list gives the monitor commands to use for each system.

System	Monitor Command
Sun-2	**g fd00000**
Sun-3	**g fd00000**
Sun386i	**g fe005000**
Sun-4	**g ffc00000**
SPARCstation 1	**go ffc00000**

The **kadb** user interface is similar to that of **adb**. Note: **kadb** prompts with

kadb>

Most **adb** commands function in **kadb** as expected. Typing an abort sequence in response to the prompt returns you to the PROM monitor, from which you can examine control spaces that are not accessible within **adb** or **kadb**. The PROM monitor command **c** will return control to **kadb**. As with '**adb –k**', **$p** works when debugging kernels (by actually mapping in new user pages). The verbs **?** and **/** are equivalent in **kadb** , since there is only one address space in use.

OPTIONS

kadb is booted from the PROM monitor as a standalone program. If you omit the **–d** flag, **kadb** automatically loads and runs **vmunix** from the filesystem **kadb** was loaded from. The **kadb vmunix** variable can be patched to change the default program to be loaded.

–d Interactive startup. Prompts with

kadb:

for a file to be loaded. From here, you can enter a boot sequence line to load a standalone program. Boot flags entered in response to this prompt are included with those already set and passed to the program. If you type a RETURN only, **kadb** loads **vmunix** from the filesystem that **kadb** was loaded from.

boot-flags

You can specify boot flags as arguments when invoking **kadb.** Note: **kadb** always sets the **–d** (debug) boot flag, and passes it to the program being debugged.

USAGE

Refer to **adb** in *Debugging Tools.*

Kernel Macros

As with **adb**, kernel macros are supported. With **kadb**, however, the macros are compiled into the debugger itself, rather than being read in from the filesystem. The **kadb** command **$M** lists macros known to **kadb.**

Setting Breakpoints

Self-relocating programs such as the SunOS kernel need to be relocated before breakpoints can be used. To set the first breakpoint for such a program, start it with '**:s**'; **kadb** is then entered after the program is relocated (when the system initializes its interrupt vectors). Thereafter, '**:s**' single-steps as with **adb**. Otherwise, use '**:c**' to start up the program.

Sun386i System Commands

The Sun386i system version of **kadb** has the following additional commands. Note, for the general syntax of **adb** commands, see **adb**(1).

:i	Read a byte (with the INB instruction) in from the port at *address*.
:o	Send a byte (with the OUTB instruction) containing *count* out through the port at *address*.
:p	Like **:b** in **adb**(1), but sets a breakpoint using the hardware debug register instead of the breakpoint instruction. The advantage of using **:p** is that when setting breakpoints with the debug register it is not necessary to have write access to the breakpoint location. Four (4) breakpoints can be set with the hardware debug registers.
$S	Switch I/O from the console to the serial port or vice versa.
[	Like **:e** in **adb**(1), but requires only one keystroke and no RETURN character.
]	Like **:s** in **adb**(1), but requires only one keystroke and no RETURN character.

Automatic Rebooting with kadb

You can set up your workstation to automatically reboot **kadb** by patching the *vmunix* variable in **/boot** with the string **kadb**. (Refer to **adb** in *Debugging Tools* for details on how to patch executables.)

FILES

/vmunix
/boot
/kadb
/usr/include/debug/debug.h

SEE ALSO

adb(1), **boot**(8S)

Debugging Tools
Writing Device Drivers

BUGS

There is no floating-point support, except on Sun386i systems.

kadb cannot reliably single-step over instructions that change the status register.

When sharing the keyboard with the operating system the monitor's input routines can leave the keyboard in a confused state. If this should happen, disconnect the keyboard momentarily and then reconnect it. This forces the keyboard to reset as well as initiating an abort sequence.

Most of the bugs listed in **adb**(1) also apply to **kadb**.

NAME

keyenvoy – talk to keyserver

SYNOPSIS

keyenvoy

DESCRIPTION

keyenvoy is used by some RPC programs to talk to the key server, **keyserv**(8C). The key server will not talk to anything but a root process, and **keyenvoy** is a set-uid root process that acts as an intermediary between a user process that wishes to talk to the key server and the key server itself.

This program cannot be run interactively.

SEE ALSO

keyserv(8C)

NAME

keyserv – server for storing public and private keys

SYNOPSIS

keyserv [**–dkn**]

DESCRIPTION

keyserv is a daemon that is used for storing the private encryption keys of each user logged into the system. These encryption keys are used for accessing secure network services such as secure NFS. When a user logs in to the system, the **login**(1) program uses the login password to decrypt the user's encryption key stored in the Network Interface Service (NIS), and then gives the decrypted key to the **keyserv** daemon to store away.

Normally, root's key is read from the file **/etc/.rootkey** when the daemon starts up. This is useful during power-failure reboots when no one is around to type a password, yet you still want the secure network services to operate normally.

OPTIONS

–d Prohibit the use of the default key. If this is used then every machine and user should have a publickey. New publickeys cannot be created if you do not already have a key. This can be done globally for an entire domain by deleting the **nobody** entry from **/etc/publickey** on the NIS master. See **chkey**(1)

–k Remember keylogins across machine reboots. This is only needed if **at**(1) is used to schedule jobs that require secure RPC. Use of this option is not recommended.

–n Do not read root's key from **/etc/.rootkey**. Instead, prompt the user for the password to decrypt **root 's** key stored in the NIS service and then store the decrypted key in **/etc/.rootkey** for future use. This option is useful if the **/etc/.rootkey** file ever gets out of date or corrupted.

FILES

/etc/.rootkey **/etc/keystore**

SEE ALSO

login(1), **keylogin**(1), **keylogout**(1), **publickey**(5)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

kgmon – generate a dump of the operating system's profile buffers

SYNOPSIS

/usr/etc/kgmon [**–bhpr**] [*filesystem*] [*memory*]

DESCRIPTION

kgmon is a tool used when profiling the operating system. When no arguments are supplied, **kgmon** indicates the state of operating system profiling as running, off, or not configured (see **config**(8)). If the **–p** flag is specified, **kgmon** extracts profile data from the operating system and produces a **gmon.out** file suitable for later analysis by **gprof**(1).

OPTIONS

–b Resume the collection of profile data.

–h Stop the collection of profile data.

–p Dump the contents of the profile buffers into a **gmon.out** file.

–r Reset all the profile buffers. If the **–p** flag is also specified, the **gmon.out** file is generated before the buffers are reset.

If neither **–b** nor **–h** is specified, the state of profiling collection remains unchanged. For example, if the **–p** flag is specified and profile data is being collected, profiling is momentarily suspended, the operating system profile buffers are dumped, and profiling is immediately resumed.

FILES

/vmunix	the default system
/dev/kmem	the default memory
gmon.out	

SEE ALSO

gprof(1), **config**(8)

DIAGNOSTICS

Users with only read permission on **/dev/kmem** cannot change the state of profiling collection. They can get a **gmon.out** file with the warning that the data may be inconsistent if profiling is in progress.

NAME

ldconfig – link-editor configuration

SYNOPSIS

/usr/etc/ldconfig [*directory* ...]

DESCRIPTION

ldconfig is used to configure a performance-enhancing cache for the run-time link-editor, **ld.so**. It is run from **/etc/rc.local** and periodically via **cron** to avoid linking with stale libraries. It should be also be run manually when a new shared object (e.g., a shared library) is installed on the system.

When invoked with no arguments, a default set of directories are built into the cache – these are the directories searched by default by the link editors. Additional directories may be specified on the command line.

FILES

/etc/ld.so.cache	holds the cached data.

SEE ALSO

ld(1)

NAME

link, unlink – exercise link and unlink system calls

SYNOPSIS

/usr/etc/link *filename1 filename2*

/usr/etc/unlink *filename*

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

link and **unlink** perform their respective system calls on their arguments, abandoning all error checking.

SEE ALSO

rm(1), **link**(2V), **unlink**(2V)

WARNINGS

Only the super-user can unlink a directory, in which case the files it contains are lost. The files can, however, be recovered from the file system's **lost+found** directory after performing an **fsck**.

If you have write permission on the directory in which *filename* resides, **unlink** removes that file without warning, regardless of its ownership.

NAME

lockd, rpc.lockd – network lock daemon

SYNOPSIS

/usr/etc/rpc.lockd [**–g** *graceperiod*] [**–t** *timeout*]

DESCRIPTION

lockd processes lock requests that are either sent locally by the kernel or remotely by another lock daemon. **lockd** forwards lock requests for remote data to the server site's lock daemon through the **rpc**(3N) **xdr**(3N) in **lockd**(8C) package. **lockd** then requests the status monitor daemon, **statd**(8C), for monitor service. The reply to the lock request will not be sent to the kernel until the status daemon and the server site's lock daemon have replied.

If either the status monitor or server site's lock daemon is unavailable, the reply to a lock request for remote data is delayed until all daemons become available.

When a server recovers, it waits for a grace period for all client site lock daemons to submit reclaim requests. Client site lock daemons, on the other hand, are notified by the status daemon of the server recovery and promptly resubmit previously granted lock requests. If **lockd** fails to secure a previously granted lock at the server site, it sends SIGLOST to a process.

OPTIONS

–t *timeout*	Use *timeout* (seconds) as the interval instead of the default value (15 seconds) to retransmit lock request to the remote server.
–g *graceperiod*	Use *graceperiod* (**seconds**) as the grace period duration instead of the default value (45 seconds).

SEE ALSO

fcntl(2V), **lockf**(3), **signal**(3V), **statd**(8C)

NAME

logintool – graphic login interface

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

logintool is started by **getty**(8) to display a full screen window for logging in. It cannot be run from the shell. It is more attractive than the traditional '**login:** ' prompt, and also provides help for the person without a username and information about the workstation.

logintool is normally invoked on the console by **getty**(8), and works only on a frame buffer.

If the **newlogin** policy in the **policies** Network Interface Service (NIS) map is set to **unrestricted**, then **logintool** may create new user accounts in the NIS service. The account resides on the local system if it is diskful, or on the system's boot server if the local system is diskless.

FILES

/usr/share/lib/ez/login

SEE ALSO

getty(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

lpc – line printer control program

SYNOPSIS

/usr/etc/lpc [*command* [*parameter*...]]

DESCRIPTION

lpc controls the operation of the printer, or of multiple printers, as described in the **/etc/printcap** database. **lpc** commands can be used to start or stop a printer, disable or enable a printer's spooling queue, rearrange the order of jobs in a queue, or display the status of each printer—along with its spooling queue and printer daemon.

With no arguments, **lpc** runs interactively, prompting with **lpc>**. If arguments are supplied, **lpc** interprets the first as a *command* to execute; each subsequent argument is taken as a *parameter* for that command. The standard input can be redirected so that **lpc** reads commands from a file.

USAGE

Commands

Commands may be abbreviated to an unambiguous substring. Note: the *printer* parameter is specified just by the name of the printer (as **lw**), not as you would specify it to **lpr**(1) or **lpq**(1) (not as **–Plw**).

? [*command*]...

help [*command*]...

Display a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort [**all** | [*printer*...]]

Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by **lpr**(1)) for the specified printers. The **abort** command can only be used by the super-user.

clean [**all** | [*printer*...]]

Remove all files with names beginning with **cf**, **tf**, or **df** from the specified printer queue(s) on the local machine. The **clean** command can only be used by the super-user.

disable [**all** | [*printer*...]]

Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by **lpr**(1). The **disable** command can only be used by the super-user.

down [**all** | [*printer*...]] [*message*]

Turn the specified printer queue off, disable printing and put *message* in the printer status file. The message doesn't need to be quoted, the remaining arguments are treated like **echo**(1V). This is normally used to take a printer down and let others know why (**lpq**(1) indicates that the printer is down, as does the **status** command).

enable [**all** | [*printer*...]]

Enable spooling on the local queue for the listed printers, so that **lpr**(1) can put new jobs in the spool queue. The **enable** command can only be used by the super-user.

exit

quit Exit from **lpc**.

restart [**all** | [*printer*...]]

Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. **lpq**(1) reports that there is no daemon present when this condition occurs. This command can be run by any user.

start [**all** | [*printer*...]]

Enable printing and start a spooling daemon for the listed printers. The **start** command can only be used by the super-user.

status [**all** | [*printer* ...]]
Display the status of daemons and queues on the local machine. This command can be run by any user.

stop [**all** | [*printer* ...]]
Stop a spooling daemon after the current job completes and disable printing. The **stop** command can only be used by the super-user.

topq *printer* [*job#* ...] [*user* ...]
Move the print job(s) specified by *job#* or those job(s) belonging to *user* to the top (head) of the printer queue. The **topq** command can only be used by the super-user.

up [**all** | [*printer* ...]] Enable everything and start a new printer daemon. Undoes the effects of **down**.

FILES

/etc/printcap	printer description file
/var/spool/*	spool directories
/var/spool/*/lock	lock file for queue control

SEE ALSO

lpq(1), **lpr**(1), **lprm**(1), **printcap**(5), **lpd**(8)

DIAGNOSTICS

?Ambiguous command
The abbreviation you typed matches more than one command.

?Invalid command
You typed a command or abbreviation that was not recognized.

?Privileged command
You used a command can be executed only by the super-user.

NAME

lpd – printer daemon

SYNOPSIS

/usr/lib/lpd [**–l**] [**–L** *logfile*] [*port#*]

DESCRIPTION

lpd is the line printer daemon (spool area handler). It is usually invoked at boot time from the **rc**(8) script, making a single pass through the **printcap**(5) file to find out about the existing printers and printing any files left after a crash. It then accepts requests to print files in a queue, transfer files to a spooling area, display a queue's status, or remove jobs from a queue. In each case, it forks a child process for each request, and continues to listen for subsequent requests.

The Internet port number used to communicate with other processes is usually obtained with **getservent**(3N), but can be specified with the *port#* argument.

If a file cannot be opened, an error message is logged using the LOG_LPR facility of **syslog**(3). **lpd** will try up to 20 times to reopen a file it expects to be there, after which it proceeds to the next file or job.

OPTIONS

–l Log valid requests received from the network. This can be useful for debugging purposes.

–L *logfile*
Change the file used for writing error conditions to *logfile*. The default is to report a message using the **syslog**(3) facility.

OPERATION

Access Control

Access control is provided by two means. First, all requests must come from one of the machines listed in either the file **/etc/hosts.equiv** or **/etc/hosts.lpd**. (This latter file is in **hosts.equiv**(5) format.) Second, if the **rs** capability is specified in the **printcap** entry, **lpr**(1) requests are only be honored for users with accounts on the printer host.

Lock File

The **lock** file in each spool directory is used to prevent multiple daemons from becoming active, and to store information about the daemon process for **lpr**(1), **lpq**(1), and **lprm**(1).

lpd uses **flock**(2) to provide exclusive access to the lock file and to prevent multiple daemons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first is the process id of the daemon and the second is the control file name of the current job being printed. The second line is updated to reflect the current status of **lpd** for the programs **lpq**(1) and **lprm**(1).

Control Files

After the daemon has successfully set the lock, it scans the directory for files beginning with **cf**. Lines in each *cf* file specify files to be printed or non-printing actions to be performed. Each such line begins with a key character that indicates what to do with the remainder of the line.

J Job name to print on the burst page.
C Classification line on the burst page.
L Literal. This line contains identification information from the password file, and causes a burst page to be printed.
T Title string for page headings printed by **pr**(1V).
H Hostname of the machine where **lpr**(1) was invoked.
P Person. Login name of the person who invoked **lpr**(1). This is used to verify ownership by **lprm**(1).
M Send mail to the specified user when the current print job completes.
f Formatted File, the name of a file to print that is already formatted.
l Like **f**, but passes control characters along, and does not make page breaks.
p Name of a file to print using **pr**(1V) as a filter.
t Troff File. The file contains **troff**(1) output (cat phototypesetter commands).

n	Ditroff File. The file contains device independent troff output.
d	DVI File. The file contains TEX output (DVI format from Stanford).
g	Graph File. The file contains data produced by **plot**(3X).
c	Cifplot File. The file contains data produced by *cifplot*.
v	The file contains a raster image.
r	The file contains text data with FORTRAN carriage control characters.
1	Troff Font R. The name of a font file to use instead of the default.
2	Troff Font I. The name of the font file to use instead of the default.
3	Troff Font B. The name of the font file to use instead of the default.
4	Troff Font S. The name of the font file to use instead of the default.
W	Width. Changes the page width (in characters) used by **pr**(1V) and the text filters.
I	Indent. Specify the number of characters by which to indent the output.
U	Unlink. The name of file to remove upon completion of printing.
N	Filename. The name of the file being printed, or a blank for the standard input (when **lpr**(1) is invoked in a pipeline).

Data Files

When a file is spooled for printing, the contents are copied into a data file in the spool directory. Data file names begin with **df**. When **lpr** is called with the **-s** option, the control files contain a symbolic link to the actual file, and no data files are created.

Minfree File

The file *minfree* in each spool directory contains the number of kilobytes to leave free so that the line printer queue won't completely fill the disk.

FILES

/etc/printcap	printer description file
/var/spool/*	spool directories
/var/spool/*/minfree	minimum free space to leave
/dev/lp*	line printer devices
/dev/printer	socket for local requests
/etc/hosts.equiv	hosts allowed equivalent host access
/etc/hosts.lpd	hosts allowed printer access only

SEE ALSO

lpq(1), **lpr**(1), **lprm**(1), **hosts**(5), **hosts.equiv**(5), **printcap**(5), **lpc**(8), **pac**(8)

NAME

mailstats – print statistics collected by sendmail

SYNOPSIS

/usr/etc/mailstats [*filename*]

DESCRIPTION

mailstats prints out the statistics collected by the **sendmail** program on mailer usage. These statistics are collected if the file indicated by the **S** configuration option of **sendmail** exists. The **mailstats** program first prints the time that the statistics file was created and the last time it was modified. It will then print a table with one row for each mailer specified in the configuration file. The first column is the mailer number, followed by the symbolic name of the mailer. The next two columns refer to the number of messages received by *sendmail*, and the last two columns refer to messages sent by *sendmail*. The number of messages and their total size (in 1024 byte units) is given. No numbers are printed if no messages were sent (or received) for any mailer.

You might want to add an entry to **/var/spool/cron/crontab/root** to reinitialize the statistics file once a night. Copy **/dev/null** into the statistics file or otherwise truncate it to reset the counters.

FILES

/etc/sendmail.st	default statistics file
/etc/sendmail.cf	sendmail configuration file
/var/spool/cron/crontab/root	
/dev/null	

SEE ALSO

sendmail(8)

BUGS

Mailstats should read the configuration file instead of having a hard-wired table mapping mailer numbers to names.

NAME

makedbm – make a NIS ndbm file

SYNOPSIS

/usr/etc/yp/makedbm [**–b**] [**–l**] [**–s**] [**–i** *yp_input_file*] [**–o** *yp_output_name*] [**–d** *yp_domain_name*] [**–m** *yp_master_name*] *infile outfile*

makedbm [**–u** *dbmfilename*]

DESCRIPTION

makedbm takes *infile* and converts it to a pair of files in **ndbm**(3) format, namely *outfile*.**pag** and *outfile*.**dir**. Each line of the input file is converted to a single **dbm** record. All characters up to the first TAB or SPACE form the key, and the rest of the line is the data. If a line ends with '\', then the data for that record is continued on to the next line. It is left for the clients of the Network Interface Service (NIS) to interpret #; **makedbm** does not itself treat it as a comment character. *infile* can be '–', in which case the standard input is read.

makedbm is meant to be used in generating **dbm** files for the NIS service, and it generates a special entry with the key *yp_last_modified*, which is the date of *infile* (or the current time, if *infile* is '–').

OPTIONS

–b Interdomain. Propagate a map to all servers using the interdomain name server **named**(8C).

–l Lowercase. Convert the keys of the given map to lower case, so that host name matches, for example, can work independent of upper or lower case distinctions.

–s Secure map. Accept connections from secure NIS networks only.

–i *yp_input_file*
Create a special entry with the key *yp_input_file*.

–o *yp_output_name*
Create a special entry with the key *yp_output_name*.

–d *yp_domain_name*
Create a special entry with the key *yp_domain_name*.

–m *yp_master_name*
Create a special entry with the key *yp_master_name*. If no master host name is specified, *yp_master_name* will be set to the local host name.

–u *dbmfilename*
Undo a **dbm** file. That is, print out a **dbm** file one entry per line, with a single space separating keys from values.

EXAMPLE

It is easy to write shell scripts to convert standard files such as **/etc/passwd** to the key value form used by **makedbm**. For example:

```
#!/bin/awk -f
BEGIN { FS = ":"; OFS = "\t"; }
{ print $1, $0 }
```

takes the **/etc/passwd** file and converts it to a form that can be read by **makedbm** to make the NIS file **passwd.byname**. That is, the key is a username, and the value is the remaining line in the **/etc/passwd** file.

SEE ALSO

yppasswd(1), **ndbm**(3), **named**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

makedev, MAKEDEV – make system special files

SYNOPSIS

/dev/MAKEDEV *device-name ...*

DESCRIPTION

MAKEDEV is a shell script normally used to install special files. It resides in the **/dev** directory, as this is the normal location of special files. Arguments to **MAKEDEV** are usually of the form *device-name?* where *device-name* is one of the supported devices listed in section 4 of the manual and '*?*' is a logical unit number (0-9). A few special arguments create assorted collections of devices and are listed below.

std Create the *standard* devices for the system; for example, **/dev/console**, **/dev/tty**.

local Create those devices specific to the local site. This request runs the shell file **/dev/MAKEDEV.local**. Site specific commands, such as those used to setup dialup lines as "ttyd?" should be included in this file.

Since all devices are created using **mknod**(8), this shell script is useful only to the super-user.

FILES

/dev/console /dev/MAKEDEV.local /dev/tty

SEE ALSO

intro(4), **config**(8), **mknod**(8)

DIAGNOSTICS

Either self-explanatory, or generated by one of the programs called from the script. Use **sh –x MAKEDEV** in case of trouble.

NAME

makekey – generate encryption key

SYNOPSIS

/usr/lib/makekey

DESCRIPTION

makekey improves the usefulness of encryption schemes depending on a key by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input, and writes 13 bytes on its standard output. The output depends on the input in a way intended to be difficult to compute (that is, to require a substantial fraction of a second).

The first eight input bytes (the *input key*) can be arbitrary ASCII characters. The last two (the *salt*) are best chosen from the set of digits, upper- and lower-case letters, and '.' and '/'. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the *output key*.

The transformation performed is essentially the following: the salt is used to select one of 4096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but modified in 4096 different ways. Using the input key as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 useful key bits in the result.

makekey is intended for programs that perform encryption (for instance, **ed**(1) and **crypt**(1)). Usually makekey's input and output will be pipes.

SEE ALSO

crypt(1), **ed**(1)

NAME

mc68881version – print the MC68881 mask number and approximate clock rate

SYNOPSIS

/usr/etc/mc68881version

AVAILABILITY

Sun-2, Sun-3, and Sun-4 systems only.

DESCRIPTION

mc68881version determines whether an MC68881 or MC68882 floating-point coprocessor is available, and if so, determines its apparent mask number and approximate clock rate and prints them on the standard output. The reported clock rate is derived by timing floating-point operations with **getrusage**(2) and is thus somewhat variable; best results may be obtained in single-user mode. The same applies to the differentiation between MC68881 and MC68882 ; these can be distinguished in user mode only by timing tests.

SEE ALSO

getrusage(2)

NAME

mconnect – connect to SMTP mail server socket

SYNOPSIS

/usr/etc/mconnect [**–p** *port*] [**–r**] [*hostname*]

DESCRIPTION

mconnect opens a connection to the mail server on a given host, so that it can be tested independently of all other mail software. If no host is given, the connection is made to the local host. Servers expect to speak the Simple Mail Transfer Protocol (SMTP) on this connection. Exit by typing the **quit** command. Typing EOF will send an end of file to the server. An interrupt closes the connection immediately and exits.

OPTIONS

–p *port* Specify the port number instead of the default SMTP port (number 25) as the next argument.

–r "Raw" mode: disable the default line buffering and input handling. This gives you a similar effect as **telnet** to port number 25, not very useful.

FILES

/usr/lib/sendmail.hf help file for SMTP commands

SEE ALSO

sendmail(8)

Postel, Jonathan B *Simple Mail Transfer Protocol*, RFC821 August 1982, SRI Network Information Center

NAME

mkfile – create a file

SYNOPSIS

mkfile [**–nv**] *size*[**k** | **b** | **m**] *filename* ...

DESCRIPTION

mkfile creates one or more files that are suitable for use as NFS-mounted swap areas, or as local swap areas. The sticky bit is set, and the file is padded with zeroes by default. The default *size* is in bytes, but it can be flagged as kilobytes, blocks, or megabytes, with the **k**, **b**, or **m** suffixes, respectively.

OPTIONS

–n Create an empty *filename*. The size is noted, but disk blocks aren't allocated until data is written to them.

–v Verbose. Report the names and sizes of created files.

SEE ALSO

swapon(2), **fstab**(5), **swapon**(8)

NAME

mkfs – construct a file system

SYNOPSIS

/usr/etc/mkfs [**–N**] *special size* [*nsect*] [*ntrack*] [*blksize*] [*fragsize*] [*ncpg*] [*minfree*] [*rps*] [*nbpi*] [*opt*] [*apc*] [*rot*] [*nrpos*]

DESCRIPTION

Note: file systems are normally created with the **newfs**(8) command.

mkfs constructs a file system by writing on the special file *special* unless the **–N** flag has been specified. *special* must be specified as a raw device and disk partition. For example, to create a file system on **sd0**, specify **/dev/rsd0[a-h]**, where **a-h** is the disk partition.

The numeric *size* specifies the number of sectors in the file system. **mkfs** builds a file system with a root directory and a lost+found directory (see **fsck**(8)). The number of inodes is calculated as a function of the file system size. No boot program is initialized by **mkfs** (see **newfs**(8)).

You must be super-user to use this command.

OPTIONS

–N Print out the file system parameters without actually creating the file system.

The following arguments allow fine tune control over the parameters of the file system.

nsect The number of sectors per track on the disk. The default is **32**.

ntrack The number of tracks per cylinder on the disk. The default is **16**.

blksize The primary block size for files on the file system. It must be a power of two, currently selected from **4096** or **8192** (the default).

fragsize The fragment size for files on the file system. The *fragsize* represents the smallest amount of disk space that will be allocated to a file. It must be a power of two currently selected from the range **512** to **8192**. The default is **1024**.

ncpg The number of disk cylinders per cylinder group. The default is **16**.

minfree The minimum percentage of free disk space allowed. Once the file system capacity reaches this threshold, only the super-user is allowed to allocate disk blocks. The default value is **10%**.

rps The rotational speed of the disk, in revolutions per second. The default is **60**.

nbpi The number of bytes for which one inode block is allocated. This parameter is currently set at one inode block for every 2048 bytes.

opt Space or time optimization preference; **s** specifies optimization for space, **t** specifies optimization for time. The default is **t**.

apc The number of alternates per cylinder (SCSI devices only). The default is **0**.

rot The expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

nrpos The number of distinguished rotational positions. The default is 8.

Users with special demands for their file systems are referred to the paper cited below for a discussion of the tradeoffs in using different configurations.

SEE ALSO

dir(5), **fs**(5), **fsck**(8), **newfs**(8), **tunefs**(8)

System and Network Administration
McKusick, Joy, Leffler; *A Fast File System for UNIX*

NOTES

newfs(8) is preferred for most routine uses.

NAME

mknod – build special file

SYNOPSIS

/usr/etc/mknod *filename* [**c**] [**b**] *major minor*

/usr/etc/mknod *filename* **p**

DESCRIPTION

mknod makes a special file. The first argument is the *filename* of the entry. In the first form, the second argument is **b** if the special file is block-type (disks, tape) or **c** if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (for example, unit, drive, or line number). Only the super-user is permitted to invoke this form of the **mknod** command.

In the second form, **mknod** makes a named pipe (FIFO).

The first form of **mknod** is only for use by system configuration people. Normally you should use **/dev/MAKEDEV** instead when making special files.

SEE ALSO

mknod(2V), **makedev**(8)

NAME

mkproto – construct a prototype file system

SYNOPSIS

/usr/etc/mkproto *special proto*

DESCRIPTION

mkproto is used to bootstrap a new file system. First a new file system is created using **newfs**(8). **mkproto** is then used to copy files from the old file system into the new file system according to the directions found in the prototype file **proto**. The prototype file contains tokens separated by SPACE or NEWLINE characters. The first tokens comprise the specification for the root directory. File specifications consist of tokens giving the mode, the user ID, the group ID, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a 6 character string. The first character specifies the type of the file. (The characters **–bcd** specify regular, block special, character special and directory files respectively.) The second character of the type is either **u** or '–' to specify set-user-id mode or not. The third is **g** or '–' for the set-group-id mode. The rest of the mode is a three digit octal number giving the owner, group, and other read, write, execute permissions, see **chmod**(1V).

Two decimal number tokens come after the mode; they specify the user and group ID's of the owner of the file.

If the file is a regular file, the next token is a pathname whence the contents and size are copied.

If the file is a block or character special file, two decimal number tokens follow which give the major and minor device numbers.

If the file is a directory, **mkproto** makes the entries '.' and '..' and then reads a list of names and (recursively) file specifications for the entries in the directory. The scan is terminated with the token **$**.

A sample prototype specification follows:

```
d--777 3 1
usr     d--777 3 1
        sh      ---755 3 1 /usr/bin/sh
        ken     d--755 6 1
                $
        b0      b--644 3 1 0 0
        c0      c--644 3 1 0 0
        $
$
```

SEE ALSO

chmod(1V), **fs**(5), **dir**(5), **fsck**(8), **newfs**(8)

BUGS

There should be some way to specify links.

There should be some way to specify bad blocks.

mkproto can only be run on virgin file systems. It should be possible to copy files into existent file systems.

NAME

modload – load a module

SYNOPSIS

modload *filename* [**–conf** *config_file*] [**–entry** *entry_point*] [**–exec** *exec_file*] [**–o** *output_file*] [**–nolink**] [**–A** *vmunix_file*]

DESCRIPTION

modload loads a loadable module into a running system. The input file *filename* is an object file (**.o** file).

OPTIONS

–conf *config_file*

Use this configuration file to configure the loadable driver being loaded. The commands in this file are the same as those that the **config**(8) program recognizes. There are two additional commands, **blockmajor** and **charmajor**, shown in the configuration file example below.

–entry *entry_point*

This is the module entry point. This is passed by **modload** to **ld**(1) when the module is linked. The default module entry point name is '*xxx* **init**'.

–exec *exec_file*

This is the name of a shell script or executable image file that is executed if the module is successfully loaded. It is always passed the module id and module type as the first two arguments. For loadable drivers, the third and fourth arguments are the block major and character major numbers respectively. For a loadable system call, the third argument is the system call number.

–o *output_file*

This is the name of the output file that is produced by the linker. If this option is omitted, then the output file name is *filename*> without the '**.o**'.

–nolink This option can be used if **modload** has already been issued once and the output file already exists. One must take care that neither the kernel nor the module have changed.

–A *vmunix_file*

This is the file that is passed to the linker to resolve module references to kernel symbols. The default is **/vmunix**. The symbol file must be for the currently running kernel or the module is likely to crash the system.

EXAMPLES

```
controller      fdc0 at atmem csr 0x001000 irq 6 priority 3
controller      fdc2 at atmem csr 0x002000 irq 5 priority 2
disk            fd0 at fdc0 drive 0
disk            fd0 at fdc0 drive 1
disk            fd0 at fdc0 drive 2
device          fd0 at fdc2 drive 0 csr 0x003000 irq 4 priority 2
disk            fd0 at fdc2 drive 1
blockmajor 51
charmajor 52
```

SEE ALSO

ld(1), **modunload**(8), **modstat**(8)

NAME

modstat – display status of loadable modules

SYNOPSIS

modstat [**–id** *module_id*]

DESCRIPTION

modstat displays the status of the loaded modules.

OPTIONS

–id *module_id*

Display the status of only this module.

SEE ALSO

modload(8), **modunload**(8)

NAME

modunload – unload a module

SYNOPSIS

modunload –id *module_id* [**–exec exec_file**]

DESCRIPTION

modunload unloads a loadable module from a running system. The *module_id* is the ID of the module as shown by **modstat**(8).

OPTIONS

–exec *exec_file*

This is the name of a shell script or executable image file that will be executed before the module is unloaded. It is always passed the module ID and module type as the first two arguments. For loadable drivers, the third and fourth arguments are the block major and character major numbers respectively. For a loadable system call, the third argument is the system call number.

SEE ALSO

modload(8), **modstat**(8)

NAME

monitor – system ROM monitor

SYNOPSIS

L1–A

BREAK

DESCRIPTION

The CPU board of the Sun workstation contains an EPROM (or set of EPROMs), called the *monitor*, that controls the system during startup. The monitor tests the system before attempting to boot the operating system. If you interrupt the boot procedure by holding down **L1** while typing **a** or **A** on the workstation keyboard (or **BREAK** if the console is a dumb terminal) the monitor issues the prompt:

>

and accepts commands interactively.

USAGE

Modes

The monitor supports three security modes (non-secure, command secure, and fully secure) and an authentication password. Access to monitor commands is controlled by these security modes. In **non-secure** mode all monitor commands are allowed. In **command secure** mode, only the **b**(boot) command with no arguments and the **c**(continue) command with no arguments may be entered without supplying the authentication password. In **fully secure** mode, only the **c**(continue) command with no arguments may be entered without supplying the authentication password. Note: The system will not auto-reboot in fully secure mode. The authentication password must be entered before booting will take place.

Commands

+|– Increment or decrement the current address and display the contents of the new location.

^C *source destination n*
(caret-C) Copy, byte-by-byte a block of length *n* from the *source* address to the *destination* address.

^I *program* (caret-I) Display the compilation date and location of *program*.

^T *virtual_address*
(caret-T) Display the physical address to which *virtual_address* is mapped.

a [*n*] [*action*]... (Sun-2 and Sun-3 systems only)
Open A–register (cpu address register) *n*, and perform indicated actions. The number *n* can be any value from **0** to **7**, inclusive. The default value is **0**. A hexadecimal *action* argument assigns the value you supply to the register *n*. A non-hex *action* terminates command input.

b [**!**] [*device* [(*c*,*u*,*p*)]] [*pathname*] [*arguments_list*]
b[?] Reset appropriate parts of the system and bootstrap a program. A '**!**' (preceding the *device* argument) prevents the system reset from occurring. Programs can be loaded from various devices (such as a disk, tape or Ethernet). '**b**' with no arguments will cause a default boot, either from a disk, or from an Ethernet controller. '**b?**' displays all boot devices and their *device* arguments, where *device* is one of:

- **ie** Intel Ethernet
- **le** Lance Ethernet (Sun-2, Sun-3, Sun-4 systems only)
- **sd** SCSI disk
- **st** SCSI 1/4" tape
- **mt** Tape Master 9-track 1/2" tape (Sun-2, Sun-3, Sun-4 systems only)
- **xd** Xylogics 7053 disk (Sun-2, Sun-3, Sun-4 systems only)
- **xt** Xylogics 1/2" tape (Sun-2, Sun-3, Sun-4 systems only)
- **xy** Xylogics 440/450 disk (Sun-2, Sun-3, Sun-4 systems only)
- **fd** Diskette (Sun386i system only)

c A controller number (**0** if only one controller),

u A unit number (**0** if only one driver), and

p A partition.

pathname A pathname for a program such as **/stand/diag**. **/vmunix** is the default.

arguments_list
A list of up to seven arguments to pass to the program being booted.

c [*virtual_address*]
Resume execution of a program. When given, *virtual_address* is the address at which execution will resume. The default is the current PC (EIP on Sun386i systems). Registers are restored to the values shown by the **a**, **d**, and **r** commands (for Sun-2 and Sun-3 systems), or by the **d** and **r** commands (for Sun-4 systems), or by the **d** command (for Sun386i systems).

d [*window_number*] (Sun-4 systems only)
Display (dump) the state of the processor. The processor state is observable only after:

- An unexpected trap was encountered.
- A user program dropped into the monitor (by calling *abortent*).
- The user manually entered the monitor by typing **L1–A** or **BREAK**.

The display consists of the following:

- The special registers: PSR, PC, nPC, TBR, WIM and Y
- Eight global registers, and
- 24 window registers (8 *in*, 8 *local*, and 8 *out*), corresponding to one of the 7 available windows. If a Floating-Point Unit is on board, its status register along with its 32 floating-point registers are also shown.

window_number
Display the indicated *window_number*, which can be any value between 0 and 6, inclusive. If no window is specified and the PSR's current window pointer contains a valid window number, registers from the window that was active just prior to entry into the monitor are displayed. Otherwise, registers from window 0 are displayed.

d (Sun386i systems only)
Display (dump) the state of the processor. This display consists of the registers, listed below:

Processor Registers:	EAX, ECX, EDX, ESI, EDI, ESP, EBP, EFLAGS, EIP
Segment Registers:	ES, CS, SS, DS, FS, GS
Memory Management Registers:	GDTR, LDTR, IDTR, TR
Control Registers:	CR0, CR2, CR3
Debug Registers:	DR0, DR1 , DR2 , DR3, DR6, DR7
Test Registers:	TR6, TR7

The processor's state is observable only after an unexpected trap, a user program has "dropped" into the monitor (by calling monitor function abortentor) or the user has manually "broken " into the monitor (by typing **L1–A** on the Workstation console, or **BREAK** on the dumb terminal's keyboard.

d [*n*] [*action*]... (Sun-2 and Sun-3 systems only)
Open **D**–register (cpu data register) *n*, and perform indicated actions. The number *n* can be any value from **0** to **7**, inclusive. The default is **0**. See the **a** command for a description of *action*.

e [*virtual_address*] [*action*] ...

Open the 16 bit word at *virtual_address* (default zero). On Sun-2, Sun-3, and Sun-4 systems, the address is interpreted in the address space defined by the **s** command. See the **a** command for a description of *action*.

f *virtual_address1 virtual_address2 pattern* [*size*] (Sun-3 and Sun-4 systems only)

Fill the bytes, words or long words from *virtual_address1* (lower) to *virtual_address2* (higher) with the constant, *pattern*. The *size* argument can take one of the following values

b byte format (the default)
w word format
l long word format

For example, the following command fills the address block from 0x1000 to 0x2000 with the word pattern, 0xABCD:

f 1000 2000 ABCD W

g [*vector*] [*argument*]
g [*virtual_address*] [*argument*]

Goto (jump to) a predetermined or default routine (first form), or to a user-specified routine (second form). The value of *argument* is passed to the routine. If the *vector* or *virtual_address* argument is omitted, the value in the PC is used as the address to jump to.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's **g** command, set the variable ***romp->v_vector_cmd** to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

%x hexadecimal
%d decimal

g0 (Sun-2, Sun-3, and Sun-4 only)

When the monitor is running as a result of the system being interrupted, force a panic and produce a crash dump.

g4 When the monitor is running as a result of the system being interrupted, force a kernel stack trace.

h (Sun-3 and Sun-4 and Sun386i systems)

Display the help menu for monitor commands and their descriptions. To return to the monitor's basic command level, press ESCAPE or **q** before pressing RETURN.

i [*cache_data_offset*] [*action*] ... (Sun-3/200 series and Sun-4 systems only)

Modify cache data RAM command. Display and/or modify one or more of the
Modify cache data RAM command. Display and/or modify one or more of the cache data addresses. See the **a** command for a description of *action*.

j [*cache_tag_offset*] [*action*] ... (Sun-3/200 series and Sun-4 systems only)

Modify cache tag RAM command. Display and/or modify the contents of one or more of the cache tag addresses. See the **a** command for a description of *action*.

k [*reset_level*]

Reset the system. If *reset_level* is:

0 CPU reset only (Sun-2 and Sun-3 systems). Reset VMEbus, interrupt registers, video monitor (Sun-4 systems). This is the default. Reset video (Sun386i systems).
1 Software reset.

2 Power-on reset. Resets and clears the memory. Runs the EPROM-based diagnostic self test, which can take several minutes, depending upon how much memory is being tested.

kb Display the system banner.

l [*virtual_address*] [*action*]...
Open the long word (32 bit) at memory address *virtual_address* (default zero). On Sun-2, Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the s command (below). See the **a** command for a description of *action*.

m [*virtual_address*] [*action*]...
Open the segment map entry that maps *virtual_address* (default zero). On Sun-2, Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the s command. Not supported on Sun386i. See the **a** command for a description of *action*.

nd (Sun386i systems only)
ne
ni Disable, enable, or invalidate the cache, respectively

o [*virtual_address*] [action]...
Open the byte location specified by
virtual_address (default zero). On Sun-2, Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the s command. See the **a** command for a description of *action*.

p [*virtual_address*] [*action*]...
Open the page map entry that maps *virtual_address* (default zero) in the address space defined by the s command. See the **a** command for a description of *action*.

p [*port_address*] [[*nonhex_char* [*hex_value*] | *hex_value*] ...] (Sun386i systems only)
Display or modify the contents of one or more port I/O addresses in byte mode. Each port address is treated as a 8-bit unit. The optional *port_address*, argument, which is a 16-bit quantity, specifies the initial port I/O address. See the **e** command for argument descriptions.

q [*eeprom_offset*] [*action*]... (Sun-3 and Sun-4 systems only)
Open the EEPROM *eeprom_offset* (default zero) in the EEPROM address space. All addresses are referenced from the beginning or base of the EEPROM in physical address space, and a limit check is performed to insure that no address beyond the EEPROM physical space is accessed. On Sun386i systems, open the NVRAM *nvram_offset* (default zero). This command is used to display or modify configuration parameters, such as: the amount of memory to test during self test, whether to display a standard or custom banner, if a serial port (A or B) is to be the system console, etc. See the **a** command for a description of *action*.

r [*reg_name*] [[*nonhex_char* [*hex_value*] | *hex_value*] ...] (Sun386i systems only)
Display or modify one or more of the processor registers. If *reg_name* is specified (2 or 3 characters from the above list), that register is displayed first. The default is EAX. See note on register availability under the command **d** (for Sun386i systems). See the **e** command for argument descriptions.

s [*step_count*] (Sun386i systems only)
Single step the execution of the interrupted program. The *step_count* argument specifies the number of single steps to execute before displaying the monitor prompt. The default is 1.

r [*register_number*] [*action*]... (Sun-2 and Sun-3 systems only)
Display and/or modify the register indicated. *register_number* can be one of:

CA 68020 Cache Address Register
CC 68020 Cache Control Register
CX 68020 System and User Context

DF Destination Function code
IS 68020 Interrupt Stack Pointer
MS 68020 Master Stack Pointer
PC Program Counter
SC 68010 System Context
SF Source Function code
SR Status Register
SS 68010 Supervisor Stack Pointer
UC 68010 User Context
US User Stack Pointer
VB Vector Base

Alterations to these registers (except **SC** and **UC**) do not take effect until the next **c** command is executed. See the **a** command for a description of *action*.

r [*register_number*] (Sun-4 systems only)
r [*register_type*]
r [**w** *window_number*]

Display and/or modify one or more of the IU or FPU registers.

A hexadecimal *register_number* can be one of:

0x00—0x0f window(0,i0)—window(0,i7), window(0,i0)—window(0,i7)
0x16—0x1f window(1,i0)—window(1,i7), window(1,i0)—window(1,i7)
0x20—0x2f window(2,i0)—window(2,i7), window(2,i0)—window(2,i7)
0x30—0x3f window(3,i0)—window(3,i7), window(3,i0)—window(3,i7)
0x40—0x4f window(4,i0)—window(4,i7), window(4,i0)—window(4,i7)
0x50—0x5f window(5,i0)—window(5,i7), window(5,i0)—window(5,i7)
0x60—0x6f window(6,i0)—window(6,i7), window(6,i0)—window(6,i7)
0x70—0x77 g0, g1, g2, g3, g4, g5, g6, g7
0x78—0x7d PSR, PC, nPC, WIM, TBR, Y
0x7e—0x9e FSR, f0—f31

Register numbers can only be displayed after an unexpected trap, a user program has entered the monitor using the *abortent* function, or the user has entered the monitor by manually typing **L1–A** or **BREAK**.

If a *register_type* is given, the first register of the indicated type is displayed. *register_type* can be one of:

f floating-point
g global
s special

If **w** and a *window_number* (**0—6**) are given, the first *in*-register within the indicated window is displayed. If *window_number* is omitted, the window that was active just prior to entering the monitor is used. If the PSR's current window pointer is invalid, window 0 is used.

s [*code*] (Sun-2 and Sun-3 systems only)

Set or query the address space to be used by subsequent memory access commands. *code* is one of:

0	undefined
1	user data space
2	user program space
3	user control space
4	undefined
5	supervisor data space
6	supervisor program space
7	supervisor control space

If *code* is omitted, **s** displays the current address space.

s [*asi*] (Sun-4 systems only)

Set or display the Address Space Identifier. With no argument, **s** displays the current Address Space Identifier. The *asi* value can be one of:

0x2	control space
0x3	segment table
0x4	Page table
0x8	user instruction
0x9	supervisor instruction
0xa	user data
0xb	supervisor data
0xc	flush segment
0xd	flush page
0xe	flush context
0xf	cache data

t [*program*] (Sun-3 systems only)

Trace the indicated standalone *program*. Works only with programs that do not affect interrupt vectors.

u [*echo*]
u [*port*] [*options*] [*baud_rate*]
u [**u**] [*virtual_address*]

With no arguments, display the current I/O device characteristics including: current input device, current output device, baud rates for serial ports A and B, an input-to-output echo indicator, and virtual addresses of mapped UART devices. With arguments, set or configure the current I/O device. With the **u** argument (**uu**...), set the I/O device to be the *virtual_address* of a UART device currently mapped.

echo Can be either **e** to enable input to be echoed to the output device, or **ne**, to indicate that input is not echoed.

port Assign the indicated *port* to be the current I/O device. *port* can be one of:

a	serial port A
b	serial port B (except on Sun386i systems)
k	the workstation keyboard
s	the workstation screen

baud_rate Any legal baud rate.

options can be any combination of:

i	input
o	output

u UART
e echo input to output
ne do not echo input
r reset indicated serial port (**a** and **b** ports only)

If either **a** or **b** is supplied, and no *options* are given, the serial port is assigned for both input and output. If **k** is supplied with no options, it is assigned for input only. If **s** is supplied with no options, it is assigned for output only.

v *virtual_address1 virtual_address2* [*size*] (Sun-3 and Sun-4 systems only)

Display the contents of *virtual_address1* (lower) *virtual_address2* (higher) in the format specified by *size*:

b byte format (the default)
w word format
l long word format

Enter return to pause for viewing; enter another return character to resume the display. To terminate the display at any time, press the space bar.

For example, the following command displays the contents of virtual address space from address 0x1000 to 0x2000 in word format:

v 1000 2000 W

w [*virtual_address*] [*argument*] (Sun-3 and Sun-4 systems only)

Set the execution vector to a predetermined or default routine. Pass *virtual_address* and *argument* to that routine.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's **w** command, set the variable ***romp->v_vector_cmd** to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

%x hexadecimal
%d decimal

x (Sun-3 and Sun-4 systems only)

Display a menu of extended tests. These diagnostics permit additional testing of such things as the I/O port connectors, video memory, workstation memory and keyboard, and boot device paths.

y c *context_number* (Sun-4 systems only)
y p|**s** *context_number virtual_address*

Flush the indicated context, context page, or context segment.

c flush context *context_number*
p flush the page beginning at *virtual_address* within context *context_number*
s flush the segment beginning at *virtual_address* within context *context_number*

z [*number*] [*breakpoint_virtual_address* [*type*] [*len*]] (Sun386i systems only)

Set or reset breakpoints for debugging. With no arguments, this command displays the existing breakpoints. The *number* argument is a values from 0 to 3, corresponding to the processor debug registers, **DR0** to **DR3**, respectively. Up to 4 distinct breakpoints can be specified. If *number* is not specified then the monitor chooses a breakpoint number. The *breakpoint_virtual_address* argument specifies the breakpoint address. The *type* argument can be one of:

- **x** Instruction Execution breakpoint (the default)
- **m** for Data Write only breakpoint
- **r** Data Reads and Writes only breakpoint.

The *len* argument can be one of: '**b**', '**w**', or '**l**', corresponding to the breakpoint field length of byte, word, or long-word, respectively. The default is '**b**'. Since the breakpoints are set in the on-chip registers, an instruction breakpoint can be placed in ROM code or in code shared by several tasks. If the *number* argument is specified but not *breakpoint_virtual_address*, the corresponding breakpoint is reset.

z [*virtual_address*] (Sun-3 systems only)

Set a breakpoint at *virtual_address* in the address space selected by the **s** command.

FILES

/vmunix

SEE ALSO

eeprom(8S)

NAME

mount, umount – mount and unmount file systems

SYNOPSIS

/usr/etc/mount [**–p**]
/usr/etc/mount –a [**fnv**] [**–t** *type*]
/usr/etc/mount [**–fnrv**] [**–t** *type*] [**–o** *options*] *filesystem directory*
/usr/etc/mount [**–vfn**] [**–o** *options*] *filesystem* | *directory*
/usr/etc/mount –d [**fnvr**] [**–o** *options*] *RFS-resource* | *directory*

/usr/etc/umount [**–t** *type*] [**–h** *host*]
/usr/etc/umount –a [**v**]
/usr/etc/umount [**–v**] *filesystem* | *directory* ...
/usr/etc/umount [**–d**] *RFS-resource* | *directory*

DESCRIPTION

mount attaches a named *filesystem* to the file system hierarchy at the pathname location *directory*, which must already exist. If *directory* has any contents prior to the **mount** operation, these remain hidden until the *filesystem* is once again unmounted. If *filesystem* is of the form *host*:*pathname*, it is assumed to be an NFS file system (type **nfs**).

umount unmounts a currently mounted file system, which can be specified either as a *directory* or a *filesystem*.

mount and **umount** maintain a table of mounted file systems in **/etc/mtab**, described in **fstab**(5). If invoked without an argument, **mount** displays the contents of this table. If invoked with either a *filesystem* or *directory* only, **mount** searches the file **/etc/fstab** for a matching entry, and mounts the file system indicated in that entry on the indicated directory.

mount also allows the creation of new, virtual file systems using **loopback mounts**. Loopback file systems provide access to existing files using alternate pathnames. Once a virtual file system is created, other file systems can be mounted within it without affecting the original file system. File systems that are subsequently mounted onto the original file system, however, **are** visible to the virtual file system, unless or until the corresponding mount point in the virtual file system is covered by a file system mounted there.

Recursive traversal of loopback mount points is not allowed; after the loopback mount of **/tmp/newroot**, the file **/tmp/newroot/tmp/newroot** does not contain yet another file system hierarchy. Rather, it appears just as **/tmp/newroot** did before the loopback mount was performed (say, as an empty directory).

The standard RC files first perform **4.2** mounts, then **nfs** mounts, during booting. On Sun386i systems, **lo** (loopback) mounts are performed just after **4.2** mounts. **/etc/fstab** files depending on alternate mount orders at boot time will fail to work as expected. Manual modification of **/etc/rc.local** will be needed to make such mount orders work.

See **lofs**(4S) and **fstab**(5) for more information and WARNINGS about loopback mounts.

OPTIONS

mount

–p Print the list of mounted file systems in a format suitable for use in **/etc/fstab**.

–a All. Attempt to mount all the file systems described in **/etc/fstab**. If a *type* argument is specified with **–t**, mount all file systems of that type. Using **–a**, **mount** builds a dependency tree of mount points in **/etc/fstab**. **mount** will correctly mount these file systems regardless of their order in **/etc/fstab** (except loopback mounts; see WARNINGS below).

–f Fake an **/etc/mtab** entry, but do not actually mount any file systems.

–n Mount the file system without making an entry in **/etc/mtab**.

–v Verbose. Display a message indicating each file system being mounted.

−t *type* Specify a file system type. The accepted types are **4.2**, **nfs**, **rfs**, **lo**, **hsfs**, and **tmp**. See **fstab**(5) for a description of **4.2**, **hsfs**, and **nfs**; see **lofs**(4S) for a description of **lo**; and see **tmpfs**(4) for a description of **tmp**. See *System and Network Administration* for details on **rfs**.

−r Mount the specified file system read-only, even if the entry in **/etc/fstab** specifies that it is to be mounted read-write.

Physically write-protected and magnetic-tape file systems must be mounted read-only. Otherwise errors occur when the system attempts to update access times, even if no write operation is attempted.

−d Mount an RFS file system. This option provides compatibility with the System V, Release 3 syntax for RFS mounts. Alternatively, the equivalent Sun syntax, **−t rfs**, may be used.

−o *options*

Specify file system *options*, a comma-separated list of words from the list below. Some options are valid for all file system types, while others apply to a specific type only.

options valid on *all* file systems:

rw|**ro**	Read/write or read-only.
suid|**nosuid**	Setuid execution allowed or disallowed.
grpid	Create files with BSD semantics for the propagation of the group ID. Under this option, files inherit the GID of the directory in which they are created, regardless of the directory's set-GID bit.
noauto	Do not mount this file system that is currently mounted read-only. If the file system is not currently mounted, an error results.
remount	If the file system is currently mounted, and if the entry in **/etc/fstab** specifies that it is to be mounted read-write or **rw** was specified along with **remount**, remount the file system making it read-write. If the entry in **/etc/fstab** specifies that it is to be mounted read-only and **rw** was not specified, the file system is not remounted. If the file system is currently mounted read-write, specifying **ro** along with **remount** results in an error. If the file system is not currently mounted, an error results.

The default is '**rw,suid**'.

options specific to **4.2** file systems:

quota|**noquota**	Usage limits are enforced, or are not enforced. The default is **noquota**.

options specific to **nfs** (NFS) file systems:

bg|**fg**	If the first attempt fails, retry in the background, or, in the foreground.
noquota	Prevent **quota**(1) from checking whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.
retry=*n*	The number of times to retry the mount operation.
rsize=*n*	Set the read buffer size to *n* bytes.
wsize=*n*	Set the write buffer size to *n* bytes.
timeo=*n*	Set the NFS timeout to *n* tenths of a second.
retrans=*n*	The number of NFS retransmissions.
port=*n*	The server IP port number.
soft|**hard**	Return an error if the server does not respond, or continue the retry request until the server responds.
intr	Allow keyboard interrupts on hard mounts.
secure	Use a more secure protocol for NFS transactions.
posix	Request POSIX.1 semantics for the file system. Requires a mount version 2 **mountd**(8C) on the server.

acregmin=*n* Hold cached attributes for at least *n* seconds after file modification.
acregmax=*n* Hold cached attributes for no more than *n* seconds after file modification.
acdirmin=*n* Hold cached attributes for at least *n* seconds after directory update.
acdirmax=*n* Hold cached attributes for no more than *n* seconds after directory update.
actimeo=*n* Set *min* and *max* times for regular files and directories to *n* seconds.
nocto Suppress fresh attributes when opening a file.
noac Suppress attribute and name (lookup) caching.

Regular defaults are:

fg,retry=10000,timeo=7,retrans=3,port=NFS_PORT,hard,
acregmin=3,acregmax=60,acdirmin=30,acdirmax=60

actimeo has no default; it sets **acregmin**, **acregmax**, **acdirmin** and **acdirmax**

Defaults for **rsize** and **wsize** are set internally by the system kernel.

options specific to **rfs** (RFS) file systems:

bg | fg If the first attempt fails, retry in the background, or, in the foreground.
retry=*n* The number of times to retry the mount operation.

Defaults are the same as for NFS.

umount

–h *host* Unmount all file systems listed in **/etc/mtab** that are remote-mounted from *host*.

–t *type* Unmount all file systems listed in **/etc/mtab** that are of a given *type*.

–a Unmount all file systems currently mounted (as listed in **/etc/mtab**).

–v Verbose. Display a message indicating each file system being unmounted.

–d Unmount an RFS file system. This option provides compatibility with the System V, Release 3 syntax for unmounting an RFS file system.

NFS FILESYSTEMS

Background vs. Foreground

Filesystems mounted with the **bg** option indicate that **mount** is to retry in the background if the server's mount daemon (mountd(8C)) does not respond. **mount** retries the request up to the count specified in the **retry=*n*** option. Once the file system is mounted, each NFS request made in the kernel waits **timeo=*n*** tenths of a second for a response. If no response arrives, the time-out is multiplied by 2 and the request is retransmitted. When the number of retransmissions has reached the number specified in the **retrans=*n*** option, a file system mounted with the **soft** option returns an error on the request; one mounted with the **hard** option prints a warning message and continues to retry the request.

Read-Write vs. Read-Only

File systems that are mounted **rw** (read-write) should use the **hard** option.

Interrupting Processes With Pending NFS Requests

The **intr** option allows keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system.

Quotas

Quota checking on NFS file systems is performed by the server, not the client; if the file system has the **quota** option on the server, quota checking is performed for both local requests and NFS requests. When a user logs in, **login**(1) runs the **quota**(1) program to check whether the user is over their quota on any of the file systems mounted on the machine. This check is performed for NFS file systems by an RPC call to the **rquotad**(8C) server on the machine from which the file system is mounted. This can be time-consuming, especially if the remote machine is down. If the **noquota** option is specified for an NFS file system, **quota** will not check whether the user is over their quota on that file system, which can speed up the process of logging in. This does *not* disable quota checking for operations on that file system; it merely disables reporting whether the user is over quota on that file system.

Secure Filesystems

The **secure** option must be given if the server requires secure mounting for the file system.

File Attributes

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting **actimeo=***n* extends flush time by *n* seconds for both regular files and directories.

SYSTEM V COMPATIBILITY

System V File-Creation Semantics

Ordinarily, when a file is created its GID is set to the effective GID of the calling process. This behavior may be overridden on a per-directory basis, by setting the set-GID bit of the parent directory; in this case, the GID is set to the GID of the parent directory (see **open**(2V) and **mkdir**(2V)). Files created on file systems that are mounted with the **grpid** option will obey BSD semantics; that is, the GID is unconditionally inherited from that of the parent directory.

EXAMPLES

To mount a local disk:

mount /dev/xy0g /usr

To fake an entry for **nd** root:

mount –ft 4.2 /dev/nd0 /

To mount all 4.2 file systems:

mount –at 4.2

To mount a remote file system:

mount –t nfs serv:/usr/src /usr/src

To mount a remote file system:

mount serv:/usr/src /usr/src

To hard mount a remote file system:

mount –o hard serv:/usr/src /usr/src

To mount an RFS remote file system, retrying in the background on failure:

mount –d –o bg SRC /usr/src

To mount an RFS remote file system read-only:

mount –t rfs –r SRC /usr/src

To save current mount state:

mount –p > /etc/fstab

Note: this is not recommended when running the automounter, see **automount**(8).

To loopback mount file systems:

mount –t lo /export/tmp/localhost /tmp
mount –t lo /export/var/localhost /var lo
mount –t lo /export/cluster/sun386.sunos4.0.1 /usr/cluster
mount –t lo /export/local/sun386 /usr/local

FILES

/etc/mtab	table of mounted file systems
/etc/fstab	table of file systems mounted at boot

WARNINGS

mount does not understand the mount order dependencies involved in loopback mounting. Loopback mounts may be dependent on two mounts having been previously performed, while **nfs** and **4.2** mounts are dependent only on a single previous mount. As a rule of thumb, place loopback mounts at the end of the **/etc/fstab** file. See **lofs**(4S) for a complete description.

SEE ALSO

mkdir(2V), **mount**(2V), **open**(2V), **unmount**(2V), **lofs**(4S), **fstab**(5), **mtab**(5), **automount**(8), **mountd**(8C), **nfsd**(8)

BUGS

Mounting file systems full of garbage crashes the system.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on *the directory to which the symbolic link refers*, rather than being mounted on top of the symbolic link itself.

NAME

mountd, rpc.mountd – NFS mount request server

SYNOPSIS

/usr/etc/rpc.mountd [**–n**]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

mountd is an RPC server that answers file system mount requests. It reads the file **/etc/xtab,** described in **exports**(5), to determine which file systems are available for mounting by which machines. It also provides information as to what file systems are mounted by which clients. This information can be printed using the **showmount**(8) command.

The **mountd** daemon is normally invoked by **rc**(8).

OPTIONS

–n Do not check that the clients are root users. Though this option makes things slightly less secure, it does allow older versions (pre-3.0) of client NFS to work.

FILES

/etc/xtab

SEE ALSO

exports(5), **rc**(8), **showmount**(8)

NAME

mount_tfs, umount_tfs – mount and dismount TFS filesystems

SYNOPSIS

/usr/etc/mount_tfs [**–r**] *fs1 fs2 ... fsN dir*
/usr/etc/mount –t tfs [**–o** *options*] *fs dir*

/usr/etc/umount_tfs *dir*
/usr/etc/umount *dir*

DESCRIPTION

mount_tfs attaches a translucent file service (TFS) filesystem to the directory *dir*. After the mount, the directory *dir* is a TFS directory whose frontmost directory is *fs1* and whose backmost directory is *dir*, with any number of directories intervening. Effectively, the directories *fs1 ... fsN* are stacked in front of *dir*.)

TFS filesystems can also be mounted using the **mount**(8) command. The **mount** command can only mount one directory, *fs*, in front of the backmost directory, *dir*.

umount_tfs detaches the TFS filesystem rooted at *dir*. See **tfs**(4S) for a description of a TFS filesystem.

OPTIONS

–r Mount the TFS filesystem read-only.

SEE ALSO

lsw(1), **unwhiteout**(1), **tfs**(4S), **mount**(8), **tfsd**(8)

BUGS

mount_tfs will cause **tfsd**(8) to deadlock (hang and answer no more requests) if it is used in conjunction with Network Software Environment (NSE) execsets. For example, a deadlock will occur if a user has used **mount_tfs** to mount over **/usr/lib**, and then tries to activate an NSE environment whose execset mounts over **/usr/lib**.

The directories *fs1*, *fs2*, ..., *fsN* must be writable.

NAME

named, in.named – Internet domain name server

SYNOPSIS

/usr/etc/in.named [**–d** *level*] [**–p** *port*] [[**–b**] *bootfile*]

DESCRIPTION

named is the Internet domain name server. It is used by resolver libraries to provide access to the Internet distributed naming database. The domain name server is described in the *System and Network Administration*. See RFC 1034 and RFC 1035 for more details. With no arguments **named** reads **/etc/named.boot** for any initial data, and listens for queries on a privileged port.

OPTIONS

–d *level* Print debugging information. *level* is a number indicating the level of messages printed.

–p *port* Use *port* as the port number, rather than the standard port number.

–b *bootfile*
Use *bootfile* rather than **/etc/named.boot**.

EXAMPLE

```
;
;          boot file for name server
;
; type              domain              source file or host
;
primary             berkeley.edu    named.db
secondary           cc.berkeley.edu 10.2.0.78 128.32.0.10
cache               .               named.ca
```

The **primary** line states that the file **named.db** contains authoritative data for **berkeley.edu**. The file **named.db** contains data in the master file format, described in RFC 1035, except that all domain names are relative to the origin; in this case, **berkeley.edu** (see below for a more detailed description).

The **secondary** line specifies that all authoritative data under **cc.berkeley.edu** is to be transferred from the name server at **10.2.0.78**. If the transfer fails it will try **128.32.0.10**, and continue for up to 10 tries at that address. The secondary copy is also authoritative for the domain.

The **cache** line specifies that data in **named.ca** is to be placed in the cache (only used to find the root domain servers). The file **named.ca** is in the same format as **named.db**.

The master file consists of entries of the form:

$INCLUDE *<filename>*
$ORIGIN *<domain>*
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>

where *domain* is '.' for the root, '@' for the current origin, or a standard domain name. If *domain* is a standard domain name that does not end with '.', the current origin is appended to the domain. Domain names ending with '.' are unmodified.

The *opt_ttl* field is an optional integer number for the time-to-live field. It defaults to zero.

The *opt_class* field is currently one token, **IN** for the Internet.

The *type* field is one of the following tokens; the data expected in the *resource_record_data* field is in parentheses.

A A host address (dotted quad).

NS An authoritative name server (domain).

MX A mail exchanger (domain).

CNAME
The canonical name for an alias (domain).

SOA Marks the start of a zone of authority (5 numbers). (see RFC 1035)).

MB A mailbox domain name (domain).

MG A mail group member (domain).

MR A mail rename domain name (domain).

NULL A null resource record (no format or data).

WKS A well know service description (not implemented yet).

PTR A domain name pointer (domain).

HINFO Host information (cpu_type OS_type).

MINFO Mailbox or mail list information (request_domain error_domain).

FILES

/etc/named.boot	name server configuration boot file
/etc/named.pid	the process ID
/var/tmp/named.run	debug output
/var/tmp/named_dump.db	dump of the name servers database

SEE ALSO

kill(1), **signal**(3V), **resolver**(3), **resolv.conf**(5), **nslookup**(8C)

System and Network Administration

Mockapetris, Paul, *Domain Names - Concepts and Facilities*, RFC 1034, Network Information Center, SRI International, Menlo Park, Calif., November 1987.

Mockapetris, Paul, *Domain Names - Implementation and Specification*, RFC 1035, Network Information Center, SRI International, Menlo Park, Calif., November 1987.

Mockapetris, Paul, *Domain System Changes and Observations*, RFC 973, Network Information Center, SRI International, Menlo Park, Calif., January 1986.

Partridge, Craig, *Mail Routing and the Domain System*, RFC 974, Network Information Center, SRI International, Menlo Park, Calif., January 1986.

NOTES

The following signals have the specified effect when sent to the server process using the **kill**(1) command.

SIGHUP Causes server to read named.boot and reload database.

SIGINT Dumps current data base and cache to **/var/tmp/named_dump.db**.

SIGUSR1

Turns on debugging; each subsequent **SIGUSR1** increments debug level.

SIGUSR2

Turns off debugging completely.

NAME

ncheck – generate names from i-numbers

SYNOPSIS

/usr/etc/ncheck [**–i** *numbers*] [**–as**] *filesystem*

DESCRIPTION

Note: For most normal file system maintenance, the function of **ncheck** is subsumed by **fsck**(8).

ncheck generates a pathname versus i-number list of files for the indicated *filesystem*. Names of directory files are followed by '.'

The report is in no useful order, and probably should be sorted.

OPTIONS

–i *numbers*
Report only those files whose i-*numbers* follow.

–a Print the names '.' and '..', which are ordinarily suppressed.

–s Report only special files and files with set-user-ID mode. This is intended to discover concealed violations of security policy.

SEE ALSO

sort(1V), **dcheck**(8), **fsck**(8), **icheck**(8)

DIAGNOSTICS

When the filesystem structure is improper, '**??**' denotes the "parent" of a parentless file and a pathname beginning with '...' denotes a loop.

NAME

ndbootd – ND boot block server

SYNOPSIS

ndbootd [**–dv**]

DESCRIPTION

ndbootd sends boot blocks to diskless Sun-2 system clients that request them using the (now obsolete) ND protocol. This server uses the boot block contained in the file **/tftpboot/sun2.bb**. A client must appear in the **ethers**(5) and **hosts**(5) databases, in order for the request to be served. In determining whether to serve the client, **ndbootd** checks the **/tftpboot** directory for a file whose name is the client's IP address in hexadecimal notation. For example, if the file **/tftpboot/C00901AD** exists, the machine at IP address 192.9.1.173 can be served. This file normally contains the boot program that is sent to the client by **tftpd**(8C).

Only root can invoke **ndbootd**.

OPTIONS

–d Debug. Display information about ignored packets, retransmissions, and address translation.

–v Verbose. Show a detailed listing of packets sent and received, etc.

If either option is used, all output is sent to the invoking terminal. Otherwise, error output (if any) appears on the console.

FILES

/tftpboot	bootfiles directory
/tftpboot/sun2.bb	boot blocks
/tftpboot/????????	boot programs for clients

SEE ALSO

ethers(5), **hosts**(5), **boot**(8S), **tftpd**(8C)

NAME

netconfig – PNP boot service

SYNOPSIS

/single/netconfig [**–e**] [**–n**]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

netconfig is used both for automatic installation of new diskful systems, and during routine booting of all systems. The sequence of actions taken by **netconfig** depends on which of these situations is in effect, but it always sets the hostname, domainname, time, timezone, and interface IP address. If the system is newly installed on the network, it does more, perhaps interrogating the user about system configuration.

netconfig is invoked with the **–e** option from the **/etc/rc.boot** script.

Invoked without options, **netconfig** may perform PNP set up, including set up of files, passwords, and secure RPCs. Unless **–n** is specified, it writes **/etc/net.conf**, which is read later by **rc.boot**. This includes the **VERBOSE** flag, derived from **NVRAM** data, which controls the verbosity of the commands in **rc.boot**.

Routine Booting

Boot servers use information stored locally in Network Interface Service (NIS) acquiring it over the network, except that they get the time from the *timehost* system if it is up. The following describes the steps taken by boot clients: diskful clients, diskless clients, and network clients.

Boot clients first invoke **rarp** to acquire an IP address. This is followed by a **ICMP Netmask** request to obtain the IP subnetwork mask, and then a **PNP_WHOAMI** RPC to determine the system's name, NIS domain, and time zone. Then the systems clock is set using the RFC 868 time service. If **PNP_WHOAMI** fails, a **PNP_SETUP** sequence is followed by set up of **/etc/passwd** and other files.

OPTIONS

–e Check shell environment variables. This option is specified during routine boot. **HOSTNAME** and **DOMAINNAME** are used to determine if the system is an NIS server using local NIS maps. Otherwise, if **NETWORKED** is **YES**, **netconfig** probes the network for network configuration. **MUST_SETUP** requires writing **/etc/passwd** and other files for setup in restricted network environments.

–n Used in conjunction with '**–e**', this does not probe the network for anything but just sets the hostname and domainname of the system from the environment variables **HOSTNAME** and **DOMAINNAME** respectively. Does not write the **/etc/net.conf** file.

FILES

/var/yp/*domainname***/netmasks**
/var/yp/*domainname***/hosts**

SEE ALSO

pnp(3R), **pnpboot**(8C), **pnpd**(8C), **rarpd**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

netstat – show network status

SYNOPSIS

netstat [**–aAn**] [**–f** *address_family*] [**system**] [**core**]

netstat [**–n**] [**–s**] [**–m** | **–i** | **–r**] [**–f** *address_family*] [**system**] [**core**]

netstat [**–n**] [**–I** *interface*] *interval* [**system**] [**core**]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

netstat displays the contents of various network-related data structures in various formats, depending on the options you select.

The first form of the command displays a list of active sockets for each protocol. The second form selects one from among various other network data structures. The third form displays running statistics of packet traffic on configured network interfaces; the *interval* argument indicates the number of seconds in which to gather statistics between displays.

The default value for the **system** argument is **/vmunix**; for *core*, the default is **/dev/kmem**.

OPTIONS

–a Show the state of all sockets; normally sockets used by server processes are not shown.

–A Show the address of any protocol control blocks associated with sockets; used for debugging.

–f *address_family*
Limit statistics or address control block reports to those of the specified *address_family*, which can be one of:

inet For the AF_INET address family, or
unix For the AF_UNIX family.

–i Show the state of interfaces that have been auto-configured. Interfaces that are statically configured into a system, but not located at boot time, are not shown.

–I *interface* Highlight information about the indicated *interface* in a separate column; the default (for the third form of the command) is the interface with the most traffic since the system was last rebooted. *interface* can be any valid interface listed in the system configuration file, such as **ie0** or **le0**.

–m Show the statistics recorded by management routines for the network's private buffer pool.

–n Show network addresses as numbers. **netstat** normally displays addresses as symbols. This option may be used with any of the display formats.

–r Show the routing tables. (When **–s** is also present, show routing statistics instead.)

–s Show per-protocol statistics. When used with the **–r** option, show routing statistics.

–t Replace queue length information with timer information.

DISPLAYS

Active Sockets (First Form)

The display for each active socket shows the local and remote address, the send and receive queue sizes (in bytes), the protocol, and the internal state of the protocol.

The symbolic format normally used to display socket addresses is either:

hostname.port

when the name of the host is specified, or:

network.port

if a socket address specifies a network but no specific host. Each *hostname* and *network* is shown according to its entry in the **/etc/hosts** or the **/etc/networks** file, as appropriate.

If the network or hostname for an address is not known (or if the **–n** option is specified), the numerical network address is shown. Unspecified, or "wildcard", addresses and ports appear as "*". (For more information regarding the Internet naming conventions, refer to **inet**(3N)).

TCP Sockets

The possible state values for TCP sockets are as follows:

CLOSED	Closed: the socket is not being used.
LISTEN	Listening for incoming connections.
SYN_SENT	Actively trying to establish connection.
SYN_RECEIVED	Initial synchronization of the connection under way.
ESTABLISHED	Connection has been established.
CLOSE_WAIT	Remote shut down: waiting for the socket to close.
FIN_WAIT_1	Socket closed, shutting down connection.
CLOSING	Closed, then remote shutdown: awaiting acknowledgement.
LAST_ACK	Remote shut down, then closed: awaiting acknowledgement.
FIN_WAIT_2	Socket closed, waiting for shutdown from remote.
TIME_WAIT	Wait after close for remote shutdown retransmission.

Network Data Structures (Second Form)

The form of the display depends upon which of the **–m**, **–i**, **–h** or **–r**, options you select. (If you specify more than one of these options, **netstat** selects one in the order listed here.)

Routing Table Display

The routing table display lists the available routes and the status of each. Each route consists of a destination host or network, and a gateway to use in forwarding packets. The *flags* column shows the status of the route (**U** if "up"), whether the route is to a gateway (**G**), and whether the route was created dynamically by a redirect (**D**).

Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface.

The **refcnt** column gives the current number of active uses per route. (Connection-oriented protocols normally hold on to a single route for the duration of a connection, whereas connectionless protocols obtain a route while sending to the same destination.)

The **use** column displays the number of packets sent per route.

The *interface* entry indicates the network interface utilized for the route.

Cumulative Traffic Statistics (Third Form)

When the *interval* argument is given, **netstat** displays a table of cumulative statistics regarding packets transferred, errors and collisions, the network addresses for the interface, and the maximum transmission unit ("mtu"). The first line of data displayed, and every 24th line thereafter, contains cumulative statistics from the time the system was last rebooted. Each subsequent line shows incremental statistics for the *interval* (specified on the command line) since the previous display.

SEE ALSO

hosts(5), **networks**(5), **protocols**(5), **services**(5) **iostat**(8), **trpt**(8C), **vmstat**(8)

BUGS

The notion of errors is ill-defined. Collisions mean something else for the IMP.

The kernel's tables can change while **netstat** is examining them, creating incorrect or partial displays.

NAME

newaliases – rebuild the data base for the mail aliases file

SYNOPSIS

newaliases

DESCRIPTION

newaliases rebuilds the random access data base for the mail aliases file **/etc/aliases**. It is run automatically by **sendmail**(8) (in the default configuration) whenever a message is sent.

FILES

/etc/aliases

SEE ALSO

aliases(5), **sendmail**(8)

NAME

newfs – create a new file system

SYNOPSIS

/usr/etc/newfs [**–Nv**] [*mkfs-options*] *raw-special-device*

DESCRIPTION

newfs is a "friendly" front-end to the **mkfs**(8) program. On Sun systems, the disk type is determined by reading the disk label for the specified *raw-special-device*.

raw-special-device is the name of a raw special device residing in **/dev**, including the disk partition, where you want the new file system to be created. If you want to make a file system on **sd0[a-h]**, specify **sd0[a-h]**, **rsd0[a-h]** or **/dev/rsd0[a-h]**; if you only specify **sd0[a-h]**, **newfs** will find the proper device.

newfs then calculates the appropriate parameters to use in calling **mkfs**, and builds the file system by forking **mkfs**.

You must be super-user to use this command.

OPTIONS

–N Print out the file system parameters without actually creating the file system.

–v Verbose. **newfs** prints out its actions, including the parameters passed to **mkfs**.

mkfs-options

Options that override the default parameters passed to **mkfs**(8) are:

–a *apc* Number of alternates per cylinder (SCSI devices only).

–b *block-size*

The block size of the file system in bytes. The default is 8192.

–c *#cylinders/group*

The number of cylinders per cylinder group in a file system. The default is 16.

–d *rotdelay*

This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

–f *frag-size*

The fragment size of the file system in bytes. The default is 1024.

–i *bytes/inode*

This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given.

–m *free-space%*

The percentage of space reserved from normal users; the minimum free space threshold. The default is 10%.

–o *optimization*

(**space** or **time**). The file system can either be instructed to try to minimize the time spent allocating blocks, or to try to minimize the space fragmentation on the disk. If the minimum free space threshold (as specified by the **–m** option) is less than 10%, the default is to optimize for **space**; if the minimum free space threshold is greater than or equal to 10%, the default is to optimize for **time**.

–r *revolutions/minute*

The speed of the disk in revolutions per minute (normally 3600).

–s *size* The size of the file system in sectors.

-t *#tracks/cylinder*
The number of tracks per cylinders on the disk. The default is 16.

-n *#rotational-positions*
The number of distinguished rotational positions. The default is 8.

EXAMPLES

The following example verbosely displays the parameters for the raw special device, **sd0a**, but does not actually create a new file system:

```
example% /usr/etc/newfs -vN sd0a
mkfs -N /dev/rsd0a 16048 34 8 8192 1024 16 10 60 2048 t 0 -1
/dev/rsd0a:     16048 sectors in 59 cylinders of 8 tracks, 34 sectors
        8.2Mb in 4 cyl groups (16 c/g, 2.23Mb/g, 896 i/g)
super-block backups (for fsck -b#) at:
 32, 4432, 8832, 13232,
example%
```

SEE ALSO

fs(5), **fsck**(8), **installboot**(8S), **mkfs**(8), **tunefs**(8)

System and Network Administration

DIAGNOSTICS

newfs: *special* **No such file or directory**
The device specified does not exist, or a disk partition was not specified.

special: **cannot open**
You must be super-user to use this command.

NOTES

To install the bootstrap programs for a root partition, run **installboot**(8S) after **newfs**.

NAME

newkey – create a new key in the publickey database

SYNOPSIS

newkey –**h** *hostname*
newkey –**u** *username*

DESCRIPTION

newkey is normally run by the network administrator on the Network Interface Service (NIS) master machine in order to establish public keys for users and super-users on the network. These keys are needed for using secure RPC or secure NFS.

newkey will prompt for the login password of the given username and then create a new public/secret key pair in **/etc/publickey** encrypted with the login password of the given user.

Use of this program is not required: users may create their own keys using **chkey**(1).

OPTIONS

–**h** *hostname* Create a new public key for the super-user at the given hostname. Prompts for the root password of the given hostname.

–**u** *username* Create a new public key for the given username. Prompts for the NIS password of the given username.

SEE ALSO

chkey(1), **keylogin**(1), **publickey**(5), **keyserv**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

nfsd, biod – NFS daemons

SYNOPSIS

/usr/etc/nfsd [*nservers*]

/usr/etc/biod [*nservers*]

DESCRIPTION

nfsd starts the daemons that handle client filesystem requests. *nservers* is the number of file system request daemons to start. This number should be based on the load expected on this server. Eight seems to be a good number.

biod starts *nservers* asynchronous block I/O daemons. This command is used on a NFS client to buffer cache handle read-ahead and write-behind. The magic number for *nservers* in here is also eight.

When a file that is opened by a client is unlinked (by the server), a file with a name of the form **.nfs***XXX* (where *XXX* is a number) is created by the client. When the open file is closed, the **.nfs***XXX* file is removed. If the client crashes before the file can be closed, the **.nfs***XXX* file is not removed.

FILES

.nfs*XXX*	client machine pointer to an open-but-unlinked file

SEE ALSO

exports(5), **mountd**(8C)

NAME

nfsstat – Network File System statistics

SYNOPSIS

nfsstat [**–cmnrsz**]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

nfsstat displays statistical information about the NFS (Network File System) and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given the default is

nfsstat –cnrs

That is, display everything, but reinitialize nothing.

OPTIONS

–c Display client information. Only the client side NFS and RPC information will be printed. Can be combined with the **–n** and **–r** options to print client NFS or client RPC information only.

–m Display statistics for each NFS mounted file system. This includes the server name and address, mount flags, current read and write sizes, the retransmission count, and the timers used for dynamic retransmission.

–n Display NFS information. NFS information for both the client and server side will be printed. Can be combined with the **–c** and **–s** options to print client or server NFS information only.

–r Display RPC information.

–s Display server information.

–z Zero (reinitialize) statistics. This option is for use by the super-user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

DISPLAYS

The server RPC display includes the fields:

calls total number of RPC calls received

badcalls total number of calls rejected

nullrecv number of times no RPC packet was available when trying to receive

badlen number of packets that were too short

xdrcall number of packets that had a malformed header

The server NFS display shows the number of NFS calls received (**calls**) and rejected (**badcalls**), and the counts and percentages for the various calls that were made.

The client RPC display includes the following fields:

calls total number of RPC calls sent
badcalls total of calls rejected by a server
retrans number of times a call had to be retransmitted
badxid number of times a reply did not match the call
timeout number of times a call timed out
wait number of times a call had to wait on a busy **CLIENT** handle
newcred number of times authentication information had to be refreshed

The client NFS display shows the number of calls sent and rejected, as well as the number of times a **CLIENT** handle was received (**nclget**), the number of times a call had to sleep while awaiting a handle (**nclsleep**), as well as a count of the various calls and their respective percentages.

FILES

/vmunix	system namelist
/dev/kmem	kernel memory

NAME

listen, nlsadmin – network listener service administration for RFS

SYNOPSIS

nlsadmin [**–mx**] [**–edr** *service_code net_spec*] [**–ikqsv** *net_spec*]
[**–lt** *addr net_spec*] [**–a** *service_code* [**–p** *modules*] **–c** *command* **–y** *comment net_spec*]
[**–qz** *code net_spec*] [**–z** *code net_spec*] [*net_spec*]

/usr/etc/listen

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

nlsadmin configures, initiates and terminates network listener (**listen**) servers for the local host. Each network (transport provider) has an associated **listen** daemon to service it locally. The **listen** daemon for each is configured separately. A **listen** daemon accepts network service requests when they arrive, and spawns servers in response to those requests. It can be used on any network (transport provider) that conforms to the transport provider specification.

nlsadmin can also report on the listener processes on a machine, either individually (per network) or collectively.

Changing the list of services provided by the listener produces immediate changes, while changing an address on which the listener listens has no effect until the listener is restarted.

nlsadmin without any options gives a brief usage message.

The *net_spec* argument to **nlsadmin** refers to a particular **listen** daemon. Specifically, *net_spec* is the relative path name of the entry under **/dev** for a given network.

–x Report the status of all of the listener processes installed on this machine.

–e *service_code net_spec*
–d *service_code net_spec*
Enable or disable, respectively, the service indicated by *service_code* for the specified network. The service must have previously been added to the listener for that network (see the **–a** option). When a listener is disabled, processes serving prior requests continue until they complete.

–r *service_code net_spec*
Remove the entry for the *service_code* from that listener's list of services.

–i *net_spec* Initialize or change a listener process for the network specified by *net_spec*. That is, create and initialize the files required by the listener. Initializing a listener with this option does not start it running. The listener must be initialized before assigning addressing or services. Note: the listener should only be initialized once for a given network.

–q *net_spec* Query the status of the listener process for the specified network. If the listener process is active, **nlsadmin** exits with a status of 0. If no such process is active, the exit code is 1. The exit code will be greater than 1 if there is an error.

–s *net_spec*
–k *net_spec*
Start or kill, respectively, the listener process for the indicated network. When a listener is killed, processes that are still running as a result of prior service requests will continue unaffected. The listener runs under its own ID of **listen** with group ID (GID) **adm**. This GID appear in the system password file **/etc/passwd**; the **HOME** directory listed for the GID is concatenated with *net_spec* to determine the location of the listener configuration information for each network.

nlsadmin may be invoked by any user to generate reports, but all operations that affect a listener's status or configuration are restricted to the super-user.

–v *net_spec* Verbose. Report on the servers associated with *net_spec*, giving the service code, status, command, and comment for each.

–l *addr net_spec* Change or set the address for the general listener service. This is the address generally used by remote processes to access the servers available through the listener (see the **–a** option). *addr* is the transport address on which to listen, and is interpreted using a syntax that allows for a variety of address formats. By default *addr* is interpreted as the symbolic ASCII representation of the transport address. An *addr* preceded by a '\x' (BACKSLASH-X) lets you enter an address in hexadecimal notation. Note: *addr* must be quoted if it contains any blanks. If *addr* is just a dash ('—'), **nlsadmin** merely reports the currently configured address.

A change of address does not take effect until the next time the listener for that network is started.

–t *addr net_spec* Change or set the address on which the listener listens for requests for terminal service. Otherwise, this is similar to **–l**. A terminal service address should not be defined unless the appropriate remote login software is available; if such software is available, it must be configured as service code **1** (see the **–a** option).

[**–m**] **–a** *service_code* **–c** *cmd* **–y** *comment net_spec*

Add a new service to the list of services available through the indicated listener. *service_code* is the code for the service, *cmd* is the command to be invoked in response to that service code, comprised of the full path name of the server and its arguments, and *comment* is a brief (free-form) description of the service for use in various reports. Note: *cmd* must be quoted if it contains arguments for the server. Similarly, *comment* must also be quoted, so as to appear to be a single word to the shell. When a service is added, it is initially enabled (see the **–e** and **–d** options).

If the **–m** option is specified, the entry is marked as an administrative entry. Service codes **1** through **100** are reserved for administrative entries, which are those that require special handling internally. In particular, code **1** is assigned to the remote login service, which is the service automatically invoked for connections to the terminal login address.

A service must explicitly be added to the listener for each network on which that service is to be available. This operation is normally performed only when the service is installed on a machine, or when populating the list of services for a new network.

–qz *code net_spec*

Query the status of the service with service code *code* on network *net_spec*, Exit with a status of 0 if the service is enabled, 1 if the service is disabled, or greater than 1 on error.

–z *code net_spec* Print a report on the server associated with *net_spec* that has service code *code*, giving the same information as in the **–v** option.

net_spec Print the status of the listener process for *net_spec*.

DIAGNOSTICS

If the command is not run under the proper ID, an error message is sent to the standard error, and the command terminates.

FILES

/usr/etc/listen
/usr/net/nls/*net_spec*

SEE ALSO

Network Programming

NAME

nslookup – query domain name servers interactively

SYNOPSIS

nslookup [**–l**] [*address*]

DESCRIPTION

nslookup is an interactive program to query Internet domain name servers. The user can contact servers to request information about a specific host or print a list of hosts in the domain.

OPTIONS

–l Use the local host's name server instead of the servers in **/etc/resolv.conf**. (If **/etc/resolv.conf** does not exist or does not contain server information, the **–l** option does not have any effect).

address Use the name server on the host machine with the given Internet address.

USAGE

Overview

The Internet domain name-space is tree-structured, with top-level domains such as:

COM commercial establishments
EDU educational institutions
GOV government agencies
MIL **MILNET** hosts

If you are looking for a specific host, you need to know something about the host's organization in order to determine the top-level domain it belongs to. For instance, if you want to find the Internet address of a machine at UCLA, do the following:

- Connect with the root server using the **root** command. The root server of the name space has knowledge of the top-level domains.
- Since UCLA is a university, its domain name is **ucla.edu**. Connect with a server for the **ucla.edu** domain with the command **server**ucla.edu. The response will print the names of hosts that act as servers for that domain. Note: the root server does not have information about **ucla.edu**, but knows the names and addresses of hosts that do. Once located by the root server, all future queries will be sent to the UCLA name server.
- To request information about a particular host in the domain (for instance, **locus**), just type the host name. To request a listing of hosts in the UCLA domain, use the **ls** command. The **ls** command requires a domain name (in this case, **ucla.edu**) as an argument.

Note: if you are connected with a name server that handles more than one domain, all lookups for host names must be fully specified with its domain. For instance, the domain **harvard.edu** is served by **seismo.css.gov**, which also services the **css.gov** and **cornell.edu** domains. A lookup request for the host **aiken** in the **harvard.edu** domain must be specified as **aiken.harvard.edu**. However, the

set *domain*=**name**

and

set defname

commands can be used to automatically append a domain name to each request.

After a successful lookup of a host, use the **finger** command to see who is on the system, or to finger a specific person. To get other information about the host, use the

set *querytype*=**value**

command to change the type of information desired and request another lookup. (**finger** requires the type to be A.)

Commands

Commands may be interrupted at any time by typing CTRL-C. To exit, type CTRL-D (EOF). The command line length must be less than 80 characters. Note: an unrecognized command will be interpreted as a host name.

host [*server*]
Look up information for *host* using the current default server or using *server* if it is specified.

server *domain*
lserver *domain*
Change the default server to *domain*. **lserver** uses the initial server to look up information about *domain* while **server** uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

root Changes the default server to the server for the root of the domain name space. Currently, the host **sri-nic.arpa** is used; this command is a synonym for '**lserver sri-nic.arpa**'.) The name of the root server can be changed with the **set root** command.

finger [*name*]
Connect with the finger server on the current host, which is defined by a previous successful lookup for a host's address information (see the **set** *querytype*=**A** command). As with the shell, output can be redirected to a named file using > and >>.

"ls [–ah]
List the information available for *domain*. The default output contains host names and their Internet addresses. The **–a** option lists aliases of hosts in the domain. The **–h** option lists CPU and operating system information for the domain. As with the shell, output can be redirected to a named file using > and >>. When output is directed to a file, hash marks are printed for every 50 records received from the server.

view*filename*
Sort and list the output of the **ls** command with **more**(1).

help

? Print a brief summary of commands.

set*keyword* [= *value*] This command is used to change state information that affects the lookups. Valid keywords are:

all Prints the current values of the various options to **set**. Information about the current default server and host is also printed.

[no]deb[ug]
Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer. The default is **nodebug**.

[no]def[name]
Append the default domain name to every lookup. The default is **nodefname.**

do[main]=*filename*
Change the default domain name to *filename*. The default domain name is appended to all lookup requests if **defname** option has been set. The default is the value in **/etc/resolv.conf**.

q[querytype]=*value*
Change the type of information returned from a query to one of:

A The host's Internet address (the default).
CNAME
The canonical name for an alias.
HINFO The host CPU and operating system type.
MD The mail destination.

MX The mail exchanger.
MB The mailbox domain name.
MG The mail group member.
MINFO The mailbox or mail list information.

(Other types specified in the RFC883 document are valid, but are not very useful.)

[no]recurse
Tell the name server to query other servers if it does not have the information. The default is **recurse.**

ret[ry]=*count*
Set the number of times to retry a request before giving up to *count*. When a reply to a request is not received within a certain amount of time (changed with **set timeout**), the request is resent. The default is *count* is **2.**

ro[ot]=*host*
Change the name of the root server to *host*. This affects the **root** command. The default root server is **sri-nic.arpa.**

t[timeout]=*interval*
Change the time-out for a reply to *interval* seconds. The default *interval is* **10** seconds.

[no]v[c]
Always use a virtual circuit when sending requests to the server. The default is **novc.**

DIAGNOSTICS

If the lookup request was not successful, an error message is printed. Possible errors are:

Time-out
The server did not respond to a request after a certain amount of time (changed with **set** *timeout*= **value**) and a certain number of retries (changed with **set** *retry*= **value**).

No information
Depending on the query type set with the **set querytype** command, no information about the host was available, though the host name is valid.

Non-existent domain
The host or domain name does not exist.

Connection refused
Network is unreachable
The connection to the name or finger server could not be made at the current time. This error commonly occurs with **finger** requests.

Server failure
The name server found an internal inconsistency in its database and could not return a valid answer.

Refused
The name server refused to service the request.

The following error should not occur and it indicates a bug in the program.

Format error
The name server found that the request packet was not in the proper format.

FILES

/etc/resolv.conf initial domain name and name server addresses.

SEE ALSO

resolver(3), **resolv.conf**(5), **named**(8C)

RFC 1034, RFC 1035

System and Network Administration

NAME

nsquery – RFS name server query

SYNOPSIS

nsquery [**–h**] [*name*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

nsquery provides information about resources available to the host from both the local domain and from other domains. All resources are reported, regardless of whether the host is authorized to access them. When used with no options, **nsquery** identifies all resources in the domain that have been advertised as sharable. A report on selected resources can be obtained by specifying *name*, where *name* is one of:

nodename	The report will include only those resources available from *nodename*.
domain.	The report will include only those resources available from *domain*.
domain.nodename	The report will include only those resources available from *domain.nodename*.

When the name does not include the delimiter '.', it will be interpreted as a *nodename* within the local domain. If the name ends with a delimiter '.', it will be interpreted as a domain name.

The information contained in the report on each resource includes its advertised name (*domain.resource*), the read/write permissions, the server (*nodename.domain*) that advertised the resource, and a brief textual description.

A remote domain must be listed in your **rfmaster** file in order to query that domain.

If no entries are found when **nsquery** is executed, the report header is printed.

If your host cannot contact the domain name server, an error message will be sent to standard error.

OPTIONS

–h Do not print header.

EXAMPLE

The following example displays the resources available from the domain **sunrfs**:

```
example% nsquery sunrfs.
RESOURCE        ACCESS      SERVER          DESCRIPTION

LOCAL_SUN3      read-only   sunrfs.estale
LOCAL_SUN4      read-only   sunrfs.estale
LOCAL_SHARE     read-only   sunrfs.estale
```

SEE ALSO

rfmaster(5), **adv**(8), **unadv**(8)

NAME

old-analyze, analyze – postmortem system crash analyzer

SYNOPSIS

/usr/old/analyze [**–dfmvD**] [**–s** *swapfile*] *corefile* [**system**]

DESCRIPTION

analyze is the post-mortem analyzer for the state of the paging system. In order to use **analyze** you must arrange to get a image of the memory (and possibly the paging area) of the system after it crashes (see **panic**(8S)).

The **analyze** program reads the relevant system data structures from the core image file and indexing information from **/vmunix** (or the specified file) to determine the state of the paging subsystem at the point of crash. It looks at each process in the system, and the resources each is using in an attempt to determine inconsistencies in the paging system state. Normally, the output consists of a sequence of lines showing each active process, its state (whether swapped in or not), its *p0br*, and the number and location of its page table pages. Any pages which are locked while raw I/O is in progress, or which are locked because they are *intransit* are also printed. (Intransit text pages often diagnose as duplicated; you will have to weed these out by hand.)

The program checks that any pages in core which are marked as not modified are, in fact, identical to the swap space copies. It also checks for non-overlap of the swap space, and that the core map entries correspond to the page tables. The state of the free list is also checked.

Options to **analyze**:

–d Print the (sorted) paging area usage.

–f Dump the free list.

–m Dump the entire coremap state.

–v (Long unused.) Use a hugely verbose output format.

–D Print the diskmap for each process.

In general, the output from this program can be confused by processes which were forking, swapping, or exiting or happened to be in unusual states when the crash occurred. You should examine the flags fields of relevant processes in the output of a **pstat**(8) to weed out such processes.

It is possible to look at the core dump with **adb**(1) if you do

adb –k /vmunix /vmcore

FILES

/vmunix default system namelist

SEE ALSO

adb(1), **ps**(1), **panic**(8S), **pstat**(8)

DIAGNOSTICS

Various diagnostics about overlaps in swap mappings, missing swap mappings, page table entries inconsistent with the core map, incore pages which are marked clean but differ from disk-image copies, pages which are locked or intransit, and inconsistencies in the free list.

It would be nice if this program analyzed the system in general, rather than just the paging system in particular.

NAME

pac – printer/plotter accounting information

SYNOPSIS

/usr/etc/pac [**–cmrs**] [**–P***printer*] [**–p***price*] [*username*...]

DESCRIPTION

pac reads the printer/plotter accounting files, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars. The accounting file is taken from the **af** field of the **printcap** entry for the printer. If any *username*s are specified, then statistics are only printed for those users; usually, statistics are printed for every user who has used any paper.

OPTIONS

–c Sort the output by cost; usually the output is sorted alphabetically by name.

–m Disregard machine names. Normally, print jobs submitted by a user from different machines would be counted separately for each machine.

–r Reverse the sorting order.

–s Summarize the accounting information on the summary accounting file. The name of the summary file is the name of the accounting file with '**_sum**' appended to it.

–P*printer* Do accounting for the named *printer*. If this option is not used, the printer specified by the PRINTER environment variable will be used if it is present; otherwise accounting is done for the default printer.

–p*price* Use the value *price* for the cost in dollars per page/foot instead of the default value of 0.02.

FILES

/etc/printcap

SEE ALSO

printcap(5)

BUGS

The relationship between the computed price and reality is as yet unknown.

NAME

panic – what happens when the system crashes

DESCRIPTION

This section explains what happens when the system crashes and how you can analyze crash dumps.

When the system crashes voluntarily, it displays a message of the form

panic: *why i gave up the ghost*

on the console, takes a dump on a mass storage peripheral, and then invokes an automatic reboot procedure as described in **reboot**(8). Unless some unexpected inconsistency is encountered in the state of the file systems due to hardware or software failure, the system will then resume multiuser operations.

The system has a large number of internal consistency checks; if one of these fails, it will panic with a very short message indicating which one failed.

When the system crashes it writes (or at least attempts to write) an image of memory into the back end of the primary swap area. After the system is rebooted, you can run the program **savecore**(8) to preserve a copy of this core image and kernel namelist for later perusal. See **savecore**(8) for details.

To analyze a dump you should begin by running **adb**(1) with the **–k** flag on the core dump, as described in *Debugging Tools*.

The most common cause of system failures is hardware failure, which can reflect itself in different ways.

See DIAGNOSTICS for some messages that you may encounter, with some hints as to causes. In each case there is a possibility that a hardware or software error produced the message in some unexpected way.

FILES

/vmunix	the system kernel
/etc/rc.local	script run when the local system starts up

SEE ALSO

adb(1), **old-analyze**(8), **reboot**(8) **sa**(8), **savecore**(8)

Debugging Tools

DIAGNOSTICS

IO err in push

hard IO err in swap The system encountered an error trying to write to the paging device or an error in reading critical information from a disk drive. You should fix your disk if it is broken or unreliable.

timeout table overflow

This really should not be a panic, but until the data structure is fixed, involved, running out of entries causes a crash. If this happens, you should make the timeout table bigger by changing the value of **ncallout** in the **param.c** file, and then rebuild your system.

trap type *type*, **pid** *process-id*, **pc** = *program-counter*, **sr** = *status-register*, **context** *context-number*

A unexpected trap has occurred within the system; typical trap types are:

- Bus error
- Address error
- Illegal instruction
- Divide by zero
- Chk instruction
- Trapv instruction
- Privilege violation
- Trace
- 1010 emulator trap
- 1111 emulator trap
- Stack format error

- Uninitialized interrupt
- Spurious interrupt

The favorite trap types in system crashes are "Bus error" or "Address error", indicating a wild reference. The *process-id* is the ID of the process running at the time of the fault, *program-counter* is the hexadecimal value of the program counter, *status-register* is the hexadecimal value of the status register, and *context-number* is the context that the process was running in. These problems tend to be easy to track down if they are kernel bugs since the processor stops cold, but random flakiness seems to cause this sometimes.

init died

The system initialization process has exited. This is bad news, as no new users will then be able to log in. Rebooting is the only fix, so the system just does it right away.

NAME

ping – send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

/usr/etc/ping *host* [*timeout*]

/usr/etc/ping [**–s**] [**–lrRv**] *host* [*packetsize*] [*count*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

ping utilizes the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from the specified *host* , or network gateway. ECHO_REQUEST datagrams, or "pings," have an IP and ICMP header, followed by a *struct*timeval, and then an arbitrary number of bytes to pad out the packet. If *host* responds, **ping** will print *host* **is alive** on the standard output and exit. Otherwise after *timeout* seconds, it will write **no answer from** *host*. The default value of *timeout* is 20 seconds.

When the **–s** flag is specified, **ping** sends one datagram per second, and prints one line of output for every ECHO_RESPONSE that it receives. No output is produced if there is no response. In this second form, **ping** computes round trip times and packet loss statistics; it displays a summary of this information upon termination or timeout. The default datagram packet size is 64 bytes, or you can specify a size with the *packetsize* command-line argument. If an optional *count* is given, **ping** sends only that number of requests.

When using **ping** for fault isolation, first '**ping**' the local host to verify that the local network interface is running.

OPTIONS

–l Loose source route. Use this option in the IP header to send the packet to the given host and back again. Usually specified with the **–R** option.

–r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to **ping** a local host through an interface that has been dropped by the router daemon, see **routed**(8C).

–R Record route. Sets the IP record route option, which will store the route of the packet inside the IP header. The contents of the record route will only be printed if the **–v** option is given, and only be set on return packets if the target host preserves the record route option across echos, or the **–l** option is given.

–v Verbose output. List any ICMP packets, other than ECHO_RESPONSE, that are received.

SEE ALSO

icmp(4P), **ifconfig**(8C), **netstat**(8C), **rpcinfo**(8C), **spray**(8C)

NAME

pnpboot, pnp.s386 – pnp diskless boot service

SYNOPSIS

/tftpboot/pnp.s386

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0*.x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

pnp.s386 is a level 2 boot program that requests actions necessary to set up a diskless workstation on the network.

The PNP diskless boot service is used by diskless workstations at installation time to locate a server that will configure the diskless client.

The last steps of the level 1 boot (from the PROM) are to load the level 2 program through **rarpd**(8C) and **tftpd**(8C). The first step in the boot sequence is RARP to acquire an IP address. This is followed by TFTP service calls to acquire the **pnp.sun*** program file needed for the client's architecture. A **PNP_ACQUIRE** RPC is then broadcast to locate a server willing to configure the diskless client.

A **PNP_SETUP** is issued to the server which returns one of three statuses: success, failure, or in_progress. As long as the server responds with a status of in_progress the client will periodically issue a **PNP_POLL** until the status changes to either success or failure.

The last step is to reboot the client. This goes through a RARP, TFTP, BOOT sequence, with the boot using the normal **boot.sun*** file and **bootparamd**(8) service.

The system will have been set up using the IP address returned in the first step and a system name will have been assigned.

FILES

/tftpboot/pnp.sun*

SEE ALSO

bootparam(3R), **bootparams**(5) **boot**(8S), **bootparamd**(8), **ipallocd**(8C), **netconfig**(8C), **pnpd**(8C), **rarpd**(8C), **tftpd**(8C)

NAME

pnpd – PNP daemon

SYNOPSIS

/usr/etc/rpc.pnpd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

pnpd is used during routine booting of systems to determine their network configuration, and by new systems to configure themselves on a network. **pnpd** adds and removes diskless clients of the boot server on which it is running. The **pnpd** daemon is normally invoked in **rc.local**. The RPCs are used by **netconfig**(8C), **pnp.s386** (see **pnpboot**(8C)), and **client**(8).

The **bootservers** Network Interface Service (NIS) map specifies limits on server capacity and default swap size.

FILES

/export/exec/*arch*
symbolic link to **/export/exec/***arch.release*

/export/exec/*arch.release*
symbolic link to **/usr** for the architecture

/export/exec/*arch.release***/boot**
root binaries

SEE ALSO

pnp(3R), **client**(8), **ipallocd**(8C), **netconfig**(8C), **pnpboot**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

portmap – TCP/IP port to RPC program number mapper

SYNOPSIS

/usr/etc/portmap

DESCRIPTION

portmap is a server that converts TCP/IP protocol port numbers into RPC program numbers. It must be running in order to make RPC calls.

When an RPC server is started, it will tell **portmap** what port number it is listening to, and what RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it will first contact **portmap** on the server machine to determine the port number where RPC packets should be sent.

Normally, standard RPC servers are started by **inetd**(8C), so **portmap** must be started before **inetd** is invoked.

SEE ALSO

inetd.conf(5), **inetd**(8C), **rpcinfo**(8C)

BUGS

If **portmap** crashes, all servers must be restarted.

NAME

praudit – print contents of an audit trail file

SYNOPSIS

praudit [**–lrs**] [**–d***del*] [*filename* ...]

AVAILABILITY

This program is available with the *Security* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

praudit reads the listed *filename*s (or standard input, if no *filename* is specified) and interprets the data as audit trail records as defined in **audit_control**(5). By default, times, security labels, user and group IDs (UIDs and GIDs, respectively) are converted to their ASCII representation. Record type and event fields are converted to long ASCII representation. A maximum of 100 audit files can be specified on the command line.

OPTIONS

–l Print records one line per record. The record type and event fields are always converted to their short ASCII representation.

–r Print records in their raw form. Times, security labels, UIDs, GIDs, record types, and events are displayed as integers. Currently, labels are not used and are displayed as zero in this mode. This option and the –s option are exclusive. If both are used, a format usage error message is output.

–s Print records in their short form. All numeric fields are converted to ASCII and displayed. The short ASCII representations for the record type and event fields are used. Security labels are displayed in their short representation. Again, labels are not currently used. This option and the **–r** option are exclusive. If both are used, a format usage error message is output.

–d*del* Use *del* as the field delimiter instead of the default delimiter, which is the comma. If *del* has special meaning for the shell, it must be quoted. The maximum size of a delimiter is four characters.

FILES

/etc/passwd

SEE ALSO

audit(2), **setuseraudit**(2), **getauditflags**(3), **audit_control**(5)

NAME

pstat – print system facts

SYNOPSIS

/usr/etc/pstat [**–afipSsT**] [**–u** *pid*] [*system* [*corefile*]]

DESCRIPTION

pstat interprets the contents of certain system tables. If *corefile* is given, the tables are sought there, otherwise in **/dev/kmem**. The required namelist is taken from **/vmunix** unless *system* is specified.

OPTIONS

–a Under **–p**, describe all process slots rather than just active ones.

–f Print the open file table with these headings:

LOC	The memory address of this table entry.
TYPE	The type of object the file table entry points to.
FLG	Miscellaneous state variables encoded thus: R open for reading W open for writing A open for appending S shared lock present X exclusive lock present I signal pgrp when data ready
CNT	Number of processes that know this open file.
MSG	Number of references from message queue.
DATA	The location of the vnode table entry or socket for this file.
OFFSET	The file offset (see **lseek**(2V)).

–i Print the inode table including the associated vnode entries with these headings:

ILOC	The memory address of this table entry.
IFLAG	Miscellaneous inode state variables encoded thus: A inode access time must be corrected C inode change time must be corrected L inode is locked R inode is being referenced U update time (**fs**(5)) must be corrected W wanted by another process (L flag is on)
IDEVICE	Major and minor device number of file system in which this inode resides.
INO	I-number within the device.
MODE	Mode bits in octal, see **chmod**(2V).
NLK	Number of links to this inode.
UID	User ID of owner.
SIZE/DEV	Number of bytes in an ordinary file, or major and minor device of special file.
VFLAG	Miscellaneous vnode state variables encoded thus: R root of its file system S shared lock applied E exclusive lock applied Z process is waiting for a shared or exclusive lock
CNT	Number of open file table entries for this vnode.
SHC	Reference count of shared locks on the vnode.
EXC	Reference count of exclusive locks on the vnode (this may be '**> 1**' if, for example, a file descriptor is inherited across a fork).
TYPE	Vnode file type, either VNON (no type), VREG (regular), VDIR (directory), VBLK (block device), VCHR (character device), VLNK (symbolic link), VSOCK (socket), VFIFO (named pipe), or VBAD (bad).

-p Print process table for active processes with these headings:

LOC — The memory address of this table entry.

S — Run state encoded thus:

0 no process
1 awaiting an event
2 (abandoned state)
3 runnable
4 being created
5 being terminated
6 stopped (by signal or under trace)

F — Miscellaneous state variables, ORed together (hexadecimal):

0000001	loaded
0000002	a system process (scheduler or page-out daemon)
0000004	locked for swap out
0000008	swapped out during process creation
0000010	process is being traced
0000020	tracing parent has been told that process is stopped
0000040	user settable lock in memory
0000080	in page-wait
0000100	prevented from swapping during **fork**(2V)
0000200	will restore old mask after taking signal
0000400	exiting
0000800	doing physical I/O
0001000	process resulted from a **vfork**(2) which is not yet complete
0002000	another flag for **vfork**(2)
0004000	process has no virtual memory, as it is a parent in the context of **vfork**(2)
0008000	process is demand paging pages from its executable image vnode
0010000	process has advised of sequential VM behavior with **vadvise**(2)
0020000	process has advised of random VM behavior with **vadvise**(2)
0080000	process is a session process group leader
0100000	process is tracing another process
0200000	process needs a profiling tick
0400000	process is scanning descriptors during select
4000000	process has done record locks
8000000	process is having its system calls traced

PRI — Scheduling priority, see **getpriority**(2).

SIG — Signals received (signals 1-32 coded in bits 0-31).

UID — Real user ID.

SLP — Amount of time process has been blocked.

TIM — Time resident in seconds; times over 127 coded as 127.

CPU — Weighted integral of CPU time, for scheduler.

NI — Nice level, see **getpriority**(2).

PGRP — Process number of root of process group.

PID — The process ID number.

PPID — The process ID of parent process.

RSS — Resident set size — the number of physical page frames allocated to this process.

SRSS — RSS at last swap (0 if never swapped).

SIZE The size of the process image. That is, the sum of the data and stack segment sizes, not including the sizes of any shared libraries.
WCHAN Wait channel number of a waiting process.
LINK Link pointer in list of runnable processes.

-S Print the streams table with these headings:

LOC The memory address of this table entry.
WRQ The address of this stream's write queue.
VNODE The address of this stream's vnode.
DEVICE Major and minor device number of device to which this stream refers.
PGRP This stream's process group number.
SIGIO The process id or process group that has this stream **open**().
FLG Miscellaneous stream state variables encoded thus:

- I waiting for **ioctl**() to finish
- R read/recvmsg is blocked
- W write/putmsg is blocked
- P priority message is at stream head
- H device has been "hung up" (**M_HANGUP**)
- O waiting for open to finish
- M stream is linked under multiplexor
- D stream is in message-discard mode
- N stream is in message-nondiscard mode
- E fatal error has occurred (**M_ERROR**)
- T waiting for queue to drain when closing
- 2 waiting for previous **ioctl**() to finish before starting new one
- 3 waiting for acknowledgment for **ioctl**()
- B stream is in non-blocking mode
- A stream is in asynchronous mode
- o stream uses old-style no-delay mode
- S stream has had **TOSTOP** set
- C VTIME clock running
- V VTIME timer expired
- r collision on **select**() for reading
- w collision on **select**() for writing
- e collision on **select**() for exceptional condition

The queues on the write and read sides of the stream are listed for each stream. Each queue is printed with these headings:

NAME The name of the module or driver for this queue.
COUNT The approximate number of bytes on this queue.
FLG Miscellaneous state variables encoded thus:

- E queue is enabled to run
- R someone wants to get from this queue when it becomes non-empty
- W someone wants to put on this queue when it drains
- F queue is full
- N queue should not be enabled automatically by a putq

MINPS The minimum packet size for this queue.
MAXPS The maximum packet size for this queue, or **INF** if there is no maximum.
HIWAT The high-water mark for this queue.
LOWAT The low-water mark for this queue.

–s Print information about swap space usage:

allocated: The amount of swap space (in bytes) allocated to private pages.

reserved: The number of swap space bytes not currently allocated, but claimed by memory mappings that have not yet created private pages.

used: The total amount of swap space, in bytes, that is either allocated or reserved.

available: The total swap space, in bytes, that is currently available for future reservation and allocation.

–T Print the number of used and free slots in the several system tables. This is useful for checking to see how full system tables have become if the system is under heavy load. Shows both used and cached inodes.

–u *pid* Print information about the process with ID *pid*.

FILES

/vmunix	namelist
/dev/kmem	default source of tables

SEE ALSO

ps(1), **chmod**(2V), **fork**(2V), **getpriority**(2), **lseek**(2V), **stat**(2V), **vadvise**(2), **vfork**(2), **fs**(5) **iostat**(8), **vmstat**(8)

BUGS

It would be very useful if the system recorded "maximum occupancy" on the tables reported by **–T**; even more useful if these tables were dynamically allocated.

NAME

pwck – check password database entries

SYNOPSIS

/usr/etc/pwck [*filename*]

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

pwck checks that a file in **passwd**(5) does not contain any errors; it checks the **/etc/passwd** file by default.

FILES

/etc/passwd

DIAGNOSTICS

Too many/few fields
An entry in the password file does not have the proper number of fields.

No login name
The login name field of an entry is empty.

Bad character(s) in login name
The login name in an entry contains characters other than lower-case letters and digits.

First char in login name not lower case alpha
The login name in an entry does not begin with a lower-case letter.

Login name too long
The login name in an entry has more than 8 characters.

Invalid UID
The user ID field in an entry is not numeric or is greater than 65535.

Invalid GID
The group ID field in an entry is not numeric or is greater than 65535.

No login directory
The login directory field in an entry is empty.

Login directory not found
The login directory field in an entry refers to a directory that does not exist.

Optional shell file not found.
The login shell field in an entry refers to a program or shell script that does not exist.

No netgroup name
The entry is a Network Interface Service (NIS) entry referring to a netgroup, but no netgroup is present.

Bad character(s) in netgroup name
The netgroup name in an NIS entry contains characters other than lower-case letters and digits.

First char in netgroup name not lower case alpha
The netgroup name in an NIS entry does not begin with a lower-case letter.

SEE ALSO

group(5), **passwd**(5)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

pwdauthd – server for authenticating passwords

SYNOPSIS

/usr/etc/rpc.pwdauthd

AVAILABILITY

This program is available with the *Security* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

pwdauthd is a server that determines authentication for users and groups. It handles authentication requests from **pwdauth**(3) and **grpauth**( ). Communication to and from **pwdauthd** is by means of RPC calls. The server is passed a *filename* and a *password*. It returns an integer value that specifies whether the *password* is valid. The possible return values are **PWA_VALID** if the name is valid, **PWA_INVALID** if the name is invalid, and **PWA_UNKNOWN** if validity cannot be determined because no adjunct files are present.

If **pwdauthd** is serving *pwdauth*, it determines whether the **passwd.adjunct** file exists. If not, it returns **PWA_UNKNOWN**. In this case, *pwdauth* knows to check the **/etc/passwd** file. Otherwise, the server calls **getpwanam**( ) (see **getpwaent**(3)) to get the entry for *filename* in either the local or the Network Interface Service (NIS) file for **passwd.adjunct**. If the encrypted password guess matches the encrypted password from the file, **pwdauthd** returns **PWA_VALID**. If the passwords do not match, it returns **PWA_INVALID**.

If **pwdauthd** is serving **grpauth**( ), it determines whether the **group.adjunct** file exists. If not, it returns **PWA_UNKNOWN**. In this case, **grpauth**( ) knows to check the **/etc/group** file. Otherwise, the server calls **getgranam**( ) (see **getgraent**(3)) to get the entry for *filename* in either the local or the NIS file for **group.adjunct**. If the encrypted password guess matches the encrypted password from the file, **pwdauthd** returns **PWA_VALID**. If the passwords do not match, it returns **PWA_INVALID**.

SEE ALSO

getgraent(3), **getpwaent**(3), **pwdauth**(3)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

quot – summarize file system ownership

SYNOPSIS

/usr/etc/quot [**–acfhnv**] [*filesystem*]

DESCRIPTION

quot displays the number of blocks (1024 bytes) in the named *filesystem* currently owned by each user.

OPTIONS

–a Generate a report for all mounted file systems.

–c Display three columns giving file size in blocks, number of files of that size, and cumulative total of blocks in that size or smaller file.

–f Display count of number of files as well as space owned by each user.

–h Estimate the number of blocks in the file — this doesn't account for files with holes in them.

–n Run the pipeline **ncheck filesystem | sort +0n | quot –n filesystem** to produce a list of all files and their owners.

–v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

FILES

/etc/mtab	mounted file systems
/etc/passwd	to get user names

SEE ALSO

du(1V), **ls**(1V)

NAME

quotacheck – file system quota consistency checker

SYNOPSIS

/usr/etc/quotacheck [**–v**] [**–p**] *filesystem*...

/usr/etc/quotacheck [**–apv**]

DESCRIPTION

quotacheck examines each file system, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated (the latter only occurs if an active file system is checked).

quotacheck expects each file system to be checked to have a quota file named *quotas* in the root directory. If none is present, **quotacheck** will ignore the file system.

quotacheck is normally run at boot time from the **/etc/rc.local** file, see **rc**(8), before enabling disk quotas with **quotaon**(8).

quotacheck accesses the raw device in calculating the actual disk usage for each user. Thus, the file systems checked should be quiescent while **quotacheck** is running.

OPTIONS

–v Indicate the calculated disk quotas for each user on a particular file system. **quotacheck** normally reports only those quotas modified.

–a Check all the file systems indicated in **/etc/fstab** to be read-write with disk quotas.

–p Run parallel passes on the required file systems, using the pass numbers in **/etc/fstab** in an identical fashion to **fsck**(8).

FILES

quotas	quota file at the file system root
/etc/mtab	mounted file systems
/etc/fstab	default file systems

SEE ALSO

quotactl(2), **quotaon**(8), **rc**(8)

NAME

quotaon, quotaoff – turn file system quotas on and off

SYNOPSIS

/usr/etc/quotaon [**–v**] *filesystem...*
/usr/etc/quotaon [**–av**]

/usr/etc/quotaoff [**–v**] *filesystem...*
/usr/etc/quotaoff [**–av**]

DESCRIPTION

quotaon

quotaon announces to the system that disk quotas should be enabled on one or more file systems. The file systems specified must be mounted at the time. The file system quota files must be present in the root directory of the specified file system and be named *quotas*.

quotaoff

quotaoff announces to the system that file systems specified should have any disk quotas turned off.

OPTIONS

quotaon

–a All file systems in **/etc/fstab** marked read-write with quotas will have their quotas turned on. This is normally used at boot time to enable quotas.

–v Display a message for each file system where quotas are turned on.

quotaoff

–a Force all file systems in **/etc/fstab** to have their quotas disabled.

–v Display a message for each file system affected.

These commands update the status field of devices located in **/etc/mtab** to indicate when quotas are on or off for each file system.

FILES

quotas	quota file at the file system root
/etc/mtab	mounted file systems
/etc/fstab	default file systems

SEE ALSO

quotactl(2), **fstab**(5), **mtab**(5)

NAME

rarpd – TCP/IP Reverse Address Resolution Protocol server

SYNOPSIS

/usr/etc/rarpd *interface* [*hostname*]

/usr/etc/rarpd –a

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rarpd starts a daemon that responds to Reverse Address Resolution Protocol (RARP) requests. The daemon forks a copy of itself that runs in background. It must be run as root.

RARP is used by machines at boot time to discover their Internet Protocol (IP) address. The booting machine provides its Ethernet Address in an RARP request message. Using the "ethers" and "hosts" databases, **rarpd** maps this Ethernet Address into the corresponding IP address which it returns to the booting machine in an RARP reply message. The booting machine must be listed in both databases for **rarpd** to locate its IP address. **rarpd** issues no reply when it fails to locate an IP address. The "ethers" and "hosts" databases may be contained either in files under **/etc** or in Network Interface Service (NIS) maps.

In the first synopsis, the *interface* parameter names the network interface upon which **rarpd** is to listen for requests. The *interface* parameter takes the "name unit" form used by **ifconfig**(8C). The second argument, *hostname*, is used to obtain the IP address of that interface. An IP address in "decimal dot" notation may be used for *hostname*. If *hostname* is omitted, the address of the interface will be obtained from the kernel. When the first form of the command is used, **rarpd** must be run separately for each interface on which RARP service is to be supported. A machine that is a router may invoke **rarpd** multiple times, for example:

/usr/etc/rarpd ie0 host
/usr/etc/rarpd ie1 host-backbone

In the second synopsis, **rarpd** locates all of the network interfaces present on the system and starts a daemon process for each one that supports RARP.

FILES

/etc/ethers
/etc/hosts

SEE ALSO

ethers(5), **hosts**(5), **policies**(5), **boot**(8S), **ifconfig**(8C), **ipallocd**(8C), **netconfig**(8C)

Finlayson, Ross, Timothy Mann, Jeffrey Mogul, and Marvin Theimer, *A Reverse Address Resolution Protocol*, RFC 903, Network Information Center, SRI International, Menlo Park, Calif., June 1984.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

rc, rc.boot, rc.local – command scripts for auto-reboot and daemons

SYNOPSIS

/etc/rc

/etc/rc.boot

/etc/rc.local

DESCRIPTION

rc and **rc.boot** are command scripts that are invoked by **init**(8) to perform file system housekeeping and to start system daemons. **rc.local** is a script for commands that are pertinent only to a specific site or client machine.

rc.boot sets the machine name, and then, if coming up multi-user, runs **fsck**(8) with the **–p** option. This "preens" the disks of minor inconsistencies resulting from the last system shutdown and checks for serious inconsistencies caused by hardware or software failure. If **fsck**(8) detects a serious disk problem, it returns an error and **init**(8) brings the system up in single-user mode. When coming up single-user, when **init**(8) is invoked by **fastboot**(8), or when it is passed the **–b** flag from **boot**(8S), functions performed in the **rc.local** file, including this disk check, are skipped.

Next, **rc** runs. If the system came up single-user, **rc** runs when the single-user shell terminates (see **init**(8)). It mounts 4.2 filesystems and spawns a shell for **/etc/rc.local**, which mounts NFS filesystems, and starts local daemons. After **rc.local** returns, **rc** starts standard daemons, preserves editor files, clears **/tmp**, starts system accounting (if applicable), starts the network (where applicable), and if enabled, runs **savecore**(8) to preserve the core image after a crash.

Sun386i

These files operate as described above with the following variations:

fsck(8) is invoked with the **–y** option to prevent users being put in single-user mode by happenstance.

rc.boot invokes **netconfig**(8C) to configure the system for the network before booting. **netconfig** is invoked before the **/usr** filesystem is mounted, because **/usr** might be mounted from a server. **netconfig** writes **/etc/net.conf** unless the **–n** option is specified, controlling system booting.

rc.boot dynamically loads device drivers.

rc invokes any programs found in **/var/recover** to clean up any operations partially completed when the system crashed or was shut down.

rc.local starts the automounter.

The file **/etc/net.conf** stores these environment variables: The **VERBOSE** environment variable controls the verbosity of the messages from the **rc** script; its value is taken from **NVRAM**. The **NETWORKED** environment variable controls whether services useful only on a networked system are started in **/etc/rc.local**. The **PNP** environment variable, set up during initial system installation, controls whether local network configuration information is used or whether that information comes from the network. (Using automatic system installation causes all systems except boot servers to get this information from the network, facilitating network reconfiguration.) The **HOSTNAME** and **DOMAINNAME** environment variables, used together, help determine if this system is a boot server or, with **PNP** set to **no**, control the host name and domain name.

FILES

/etc/rc
/etc/rc.boot
/etc/rc.local
/etc/net.conf
/var/recover/*
/var/yp/*
/tmp

SEE ALSO

automount(8), **boot**(8S), **fastboot**(8), **init**(8), **reboot**(8), **savecore**(8), **netconfig**(8C)

BUGS

The system message file **/var/adm/messages** is no longer created automatically.

NAME

rdate – set system date from a remote host

SYNOPSIS

/usr/ucb/rdate *hostname*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rdate sets the local date and time from the *hostname* given as argument. You must be super-user on the local system. Typically **rdate** can be inserted as part of your **/etc/rc.local** startup script.

FILES

/etc/rc.local

BUGS

Could be modified to accept a list of hostnames and try each until a valid date returned. Better yet would be to write a real date server that accepted broadcast requests.

NAME

reboot – restart the operating system

SYNOPSIS

/usr/etc/reboot [**–dnq**] [*boot arguments*]

DESCRIPTION

reboot executes the **reboot**(2) system call to restart the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to it. See **boot**(8S) for details.

Although **reboot** can be run by the super-user at any time, **shutdown**(8) is normally used first to warn all users logged in of the impending loss of service. See **shutdown**(8) for details.

reboot performs a **sync**(1) operation on the disks, and then a multiuser reboot is initiated. See **init**(8) for details.

reboot normally logs the reboot to the system log daemon, **syslogd**(8), and places a shutdown record in the login accounting file **/var/adm/wtmp**. These actions are inhibited if the **–n** or **–q** options are present.

Power Fail and Crash Recovery

Normally, the system will reboot itself at power-up or after crashes.

OPTIONS

–d Dump system core before rebooting.

–n Avoid the **sync**(1). It can be used if a disk or the processor is on fire.

–q Quick. Reboots quickly and ungracefully, without first shutting down running processes.

Boot Arguments

If a boot argument string is given, it is passed to the boot command in the PROM monitor. The string must be quoted if it contains spaces or other characters that could be interpreted by the shell. If the first character of the boot argument string is a minus sign '–' the string must be preceded by an option terminator string '––' For example: 'reboot – – –s' to reboot and come up single user, '**reboot vmunix.test**' to reboot to a new kernel. See **boot**(8S) for details.

FILES

/var/adm/wtmp login accounting file

SEE ALSO

sync(1), **reboot**(2), **boot**(8S), **fsck**(8), **halt**(8), **init**(8), **panic**(8S), **shutdown**(8), **syslogd**(8)

NAME

renice – alter nice value of running processes

SYNOPSIS

/usr/etc/renice *priority pid...*

/usr/etc/renice *priority* [**–p** *pid...*] [**–g** *pgrp...*] [**–u** *username...*]

DESCRIPTION

renice alters the scheduling nice value, and hence the priority, of one or more running processes. See **nice**(1) for a discussion of nice value and process scheduling priority.

OPTIONS

By default, the processes to be affected are specified by their process IDs. *priority* is the new priority value.

–p *pid...* Specify a list of process IDs.

–g *pgrp...* Specify a list of process group IDs. The processes in the specified process groups have their scheduling priority altered.

–u *user...* Specify a list of user IDs or usernames. All processes owned by each *user* have their scheduling altered.

Users other than the super-user may only alter the priority of processes they own, and can only monotonically increase their "nice value" within the range 0 to 20. (This prevents overriding administrative fiats.) The super-user may alter the priority of any process and set the priority to any value in the range –20 to 19. Useful nice values are 19 (the affected processes will run only when nothing else in the system wants to), 0 (the default nice value) and any negative value (to make things go faster).

If only the priority is specified, the current process (alternatively, process group or user) is used.

FILES

/etc/passwd to map user names to user ID's

SEE ALSO

pstat(8)

BUGS

If you make the nice value very negative, then the process cannot be interrupted.

To regain control you must make the priority greater than zero.

Users other than the super-user cannot increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

NAME

repquota – summarize quotas for a file system

SYNOPSIS

/usr/etc/repquota [**–v**] *filesystem*...

/usr/etc/repquota [**–av**]

DESCRIPTION

repquota prints a summary of the disc usage and quotas for the specified file systems. For each user the current number of files and amount of space (in kilobytes) is printed, along with any quotas created with **edquota**(8).

OPTIONS

–a Report on all file systems indicated in **/etc/fstab** to be read-write with quotas.

–v Report all quotas, even if there is no usage.

Only the super-user may view quotas which are not their own.

FILES

quotas	quota file at the file system root
/etc/fstab	default file systems

SEE ALSO

quota(1), **quotactl**(2), **edquota**(8), **quotacheck**(8), **quotaon**(8)

NAME

restore, rrestore – incremental file system restore

SYNOPSIS

/usr/etc/restore –irRtx [*filename* ...]

DESCRIPTION

restore restores files from backup tapes created with the **dump**(8) command. *options* is a string of at least one of the options listed below, along with any modifiers and arguments you supply. Remaining arguments to **restore** are the names of files (or directories whose files) are to be restored to disk. Unless the **h** modifier is in effect, a directory name refers to the files it contains, and (recursively) its subdirectories and the files they contain.

OPTIONS

i Interactive. After reading in the directory information from the tape, **restore** invokes an interactive interface that allows you to browse through the dump tape's directory hierarchy, and select individual files to be extracted. See **Interactive Commands**, below, for a description of available commands.

r Restore the entire tape. Load the tape's full contents into the current directory. This option should only be used to restore a complete dump tape onto a clear filesystem, or to restore an incremental dump tape after a full "level **0**" restore. For example:

```
example# /usr/etc/newfs /dev/rxy0g
example# /usr/etc/mount /dev/xy0g /mnt
example# cd /mnt
example# restore r
```

is a typical sequence to restore a "level **0**" dump. Another **restore** can be done to get an incremental dump in on top of this.

R Resume restoring. **restore** requests a particular tape of a multivolume set from which to resume a full restore (see the **r** option above). This allows **restore** to start from a checkpoint when it is interrupted in the middle of a full restore.

t Table of contents. List each *filename* that appears on the tape. If no *filename* argument is given, the root directory is listed. This results in a list of all files on the tape, unless the **h** modifier is in effect. (The **t** option replaces the function of the old **dumpdir** program).

x Extract the named files from the tape. If a named file matches a directory whose contents were written onto the tape, and the **h** modifier is not in effect, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no *filename* argument is given, the root directory is extracted. This results in the entire tape being extracted unless the **h** modifier is in effect.

Modifiers

Some of the following modifiers take arguments that are given as separate words on the command line. When more than one such modifier appears within *options*, the arguments must appear in the same order as the modifiers that they apply to.

a *archive-file*

The dump table of contents is taken from the specified *archive-file* instead of from a dump tape. If a requested file is present in the table of contents, *restore* will prompt for the tape volume to be mounted. If only contents information is needed, for example when the *t* option is specified, or the *i* option is specified without a corresponding *extract* request, no dump tape will have to be mounted.

c Convert the contents of the dump tape to the new filesystem format.

d Debug. Turn on debugging output.

h Extract the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

m Extract by inode numbers rather than by filename to avoid regenerating complete pathnames. This is useful if only a few files are being extracted.

v Verbose. **restore** displays the name of each file it restores, preceded by its file type.

y Do not ask whether to abort the restore in the event of tape errors. **restore** tries to skip over the bad tape block(s) and continue as best it can.

b *factor*
Blocking factor. Specify the blocking factor for tape reads. By default, **restore** will attempt to figure out the block size of the tape. Note: a tape block is 512 bytes.

f *dump-file*
Use *dump-file* instead of **/dev/rmt?** as the file to restore from. If *dump-file* is specified as '–', **restore** reads from the standard input. This allows, **dump**(8) and **restore** to be used in a pipeline to dump and restore a file system:

example# dump 0f – /dev/rxy0g | (cd /mnt; restore xf –)

If the name of the file is of the form *machine*:*device* the restore is done from the specified machine over the network using **rmt**(8C). Since **restore** is normally run by root, the name of the local machine must appear in the **.rhosts** file of the remote machine. If the file is specified as *user*@*machine*:*device*, **restore** will attempt to execute as the specified user on the remote machine. The specified user must have a **.rhosts** file on the remote machine that allows root from the local machine.

s *n* Skip to the *n*'th file when there are multiple dump files on the same tape. For example, the command:

example# restore xfs /dev/nrar0 5

would position you at the fifth file on the tape.

USAGE

Interactive Commands

restore enters interactive mode when invoked with the **i** option. Interactive commands are reminiscent of the shell. For those commands that accept an argument, the default is the current directory.

ls [*directory*]
List files in *directory* or the current directory, represented by a '.' (period). Directories are appended with a '/' (slash). Entries marked for extraction are prefixed with a '*' (asterisk). If the verbose option is in effect, inode numbers are also listed.

cd *directory*
Change to directory *directory* (within the dump-tape).

pwd Print the full pathname of the current working directory.

add [*filename*]
Add the current directory, or the named file or directory **directory** to the list of files to extract. If a directory is specified, add that directory and its files (recursively) to the extraction list (unless the **h** modifier is in effect).

delete [*filename*]
Delete the current directory, or the named file or directory from the list of files to extract. If a directory is specified, delete that directory and all its descendents from the extraction list (unless the **h** modifier is in effect). The most expedient way to extract a majority of files from a directory is to add that directory to the extraction list, and then delete specific files to omit.

extract Extract all files on the extraction list from the dump tape. **restore** asks which volume the user wishes to mount. The fastest way to extract a small number of files is to start with the last tape volume and work toward the first.

verbose Toggle the status of the **v** modifier. While **v** is in effect, the **ls** command lists the inode numbers of all entries, and **restore** displays information about each file as it is extracted.

help Display a summary of the available commands.

quit **restore** exits immediately, even if the extraction list is not empty.

FILES

/dev/rmt8	the default tape drive
dumphost:/dev/rmt8	the default tape drive if called as **rrestore**
/tmp/rstdir*	file containing directories on the tape
/tmp/rstmode*	owner, mode, and timestamps for directories
./restoresymtable	information passed between incremental restores

SEE ALSO

dump(8), **mkfs**(8), **mount**(8), **newfs**(8), **rmt**(8C)

DIAGNOSTICS

restore complains about bad option characters.

Read errors result in complaints. If **y** has been specified, or the user responds **y**, **restore** will attempt to continue.

If the dump extends over more than one tape, **restore** asks the user to change tapes. If the **x** or **i** option has been specified, **restore** also asks which volume the user wishes to mount.

There are numerous consistency checks that can be listed by **restore**. Most checks are self-explanatory or can "never happen". Common errors are given below.

Converting to new file system format.
A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

filename: **not found on tape**
The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file *inumber*, **got** *inumber*
A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

Incremental tape too low
When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

Incremental tape too high
When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or one that has too high an incremental level has been loaded.

Tape read error while restoring *filename*
Tape read error while skipping over inode inumber
Tape read error while trying to resynchronize
A tape read error has occurred.
If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

resync restore, skipped *num* **blocks**

After a tape read error, **restore** may have to resynchronize itself. This message lists the number of blocks that were skipped over.

BUGS

restore can get confused when doing incremental restores from dump tapes that were made on active file systems.

A "level **0**" dump must be done after a full restore. Because **restore** runs in user mode, it has no control over inode allocation; this means that **restore** repositions the files, although it does not change their contents. Thus, a full dump must be done to get a new set of directories reflecting the new file positions, so that later incremental dumps will be correct.

NAME

rexd, rpc.rexd – RPC-based remote execution server

SYNOPSIS

/usr/etc/rpc.rexd [–s]

DESCRIPTION

rexd is the Sun RPC server for remote program execution. This daemon is started by **inetd**(8C) whenever a remote execution request is made.

For noninteractive programs, the standard file descriptors are connected directly to TCP connections. Interactive programs involve pseudo-terminals, in a fashion that is similar to the login sessions provided by **rlogin**(1C). This daemon may use NFS to mount file systems specified in the remote execution request.

FILES

/dev/ttyp*n*	pseudo-terminals used for interactive mode
/etc/passwd	authorized users
/tmp_rex/rexd??????	temporary mount points for remote file systems.

OPTIONS

–s Secure. When specified, requests must have valid des credentials. If the request does not have a DES credential it is rejected. The default publickey credential is rejected. Only newer **on** commands send DES credentials.

If access is denied with an Authentication error, you may have to set your publickey with the **chkey**(1) command.

SEE ALSO

chkey(1), **on**(1C), **rlogin**(1C), **rex**(3R), **exports**(5), **inetd.conf**(5), **publickey**(5), **inetd**(8C)

DIAGNOSTICS

Diagnostic messages are normally printed on the console, and returned to the requestor.

RESTRICTIONS

Root cannot execute commands using **rexd** client programs such as **on**(1C).

NAME

rexecd, in.rexecd – remote execution server

SYNOPSIS

/usr/etc/in.rexecd *host.port*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rexecd is the server for the **rexec**(3N) routine. The server provides remote execution facilities with authentication based on user names and encrypted passwords. It is invoked automatically as needed by **inetd**(8C), and then executes the following protocol:

- The server reads characters from the socket up to a null (**\0**) byte. The resultant string is interpreted as an ASCII number, base 10.
- If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.
- A null terminated user name of at most 16 characters is retrieved on the initial socket.
- A null terminated, encrypted, password of at most 16 characters is retrieved on the initial socket.
- A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
- **rexecd** then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.
- A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rexecd**.

SEE ALSO

rexec(3N) **inetd**(8C)

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

username too long
The name is longer than 16 characters.

password too long
The password is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect.
No password file entry for the user name existed.

Password incorrect.
The wrong password was supplied.

No remote directory.
The **chdir** command to the home directory failed.

Try again.
A **fork** by the server failed.

/usr/bin/sh: ...

The user's login shell could not be started.

BUGS

Indicating **'Login incorrect'** as opposed to **'Password incorrect'** is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data exchanges to be encrypted should be present.

NAME

rfadmin – RFS domain administration

SYNOPSIS

rfadmin
rfadmin –p
rfadmin –a *hostname*
rfadmin –r *hostname*

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rfadmin is used to add and remove hosts and their associated authentication information from a *domain*/**passwd** file on a Remote File Sharing (RFS) primary domain name server. It is also used to transfer domain name server responsibilities from one machine to another. Used with no options, **rfadmin** returns the *hostname* of the current domain name server for the local domain. For each *domain*, **/usr/nserve/auth.info/***domain*/**passwd** is created on the primary, and should be copied to all secondaries, and all hosts that want to do password verification of hosts in the *domain*.

rfadmin can only be used to modify domain files on the primary domain name server (**–a** and **–r** options). If domain name server reponsibilities are temporarily passed to a secondary domain name server, that computer can use the **–p** option to pass domain name server responsibility back to the primary. Any host can use **rfadmin** with no options to print information about the domain. The user must have **root** permissions to use the command.

Using **rfadmin** with the **–a** option, will result in an error if *hostname* is not unique in the domain.

Using **rfadmin** with the **–r** option, will send an error to the standard error if one of the following is true:

- *hostname* does not exist in the domain.
- *hostname* is defined as a domain name server.
- There are resources advertised by *hostname*.

When used with the **–p** option, **rfadmin** sends an error message to standard error, if there are no backup name servers defined for *domain*.

OPTIONS

–p Pass the domain name server responsibilities back to a primary or to a secondary name server.

–a *hostname*
Add a host to a domain that is served by this domain name server. *hostname* must be of the form *domain.nodename*. Create an entry for *hostname* in the *domain*/**passwd** file, which has the same format as **/etc/passwd**, and prompt for an initial authentication password; the password prompting process conforms with that of **passwd**(1).

–r *hostname*
Remove a host from its domain by removing it from the *domain*/**passwd** file.

FILES

/usr/nserve/auth.info/*domain*/**passwd**

SEE ALSO

passwd(1), **mount**(8), **rfstart**(8), **rfstop**(8)

NAME

rfpasswd – change RFS host password

SYNOPSIS

rfpasswd

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rfpasswd updates the Remote File Sharing (RFS) authentication password for a host; processing of the new password follows the same criteria as **passwd**(1). The updated password is registered at the domain name server (**/usr/nserve/auth.info/***domain***/passwd**) and replaces the password stored at the local host (**/usr/nserve/loc.passwd/file**).

This command is restricted to the super-user.

Note: if you change your host password, make sure that hosts that validate your password are notified of this change. To receive the new password, hosts must obtain a copy of the *domain/passwd* file from the domain's primary name server. If this is not done, *attempts to mount remote resources may fail*.

If any of the following is true an error message will be sent to the standard error:

- The old password entered from this command does not match the existing password for this machine.
- The two new passwords entered from this command do not match.
- The new password does not satisfy the security criteria in **passwd**(1).
- The domain name server does not know about this machine.
- The command is not run with super-user privileges.

Also, RFS must be running on your host and your domain's primary name server. A new password cannot be logged if a secondary is acting as the domain name server.

FILES

/usr/nserve/auth.info/*domain***/passwd**
/usr/nserve/loc.passwd

SEE ALSO

passwd(1), **rfadmin**(8), **rfstart**(8)

NAME

rfstart – start RFS

SYNOPSIS

rfstart [**–v**] [**–p** *primary_addr*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rfstart starts Remote File Sharing (RFS) and defines an authentication level for incoming requests. This command can be used only after the domain name server is set up and your computer's domain name and network specification has been defined using **dname**(8).

If the host password has not been set, **rfstart** will prompt for a password; the password prompting process must match the password entered for your machine at the primary domain name server (see **rfadmin**(8)). If you remove the **loc.passwd** file or change domains, you will also have to reenter the password.

Also, when **rfstart** is run on a domain name server, entries in the **rfmaster**(5) file are syntactically validated.

This command is restricted to the super-user.

If syntax errors are found in validating the **rfmaster**(5) file, a warning describing each error will be sent to the standard error.

An error message will be sent to the standard error if any of the following is true:

- The shared resource environment is already running.
- There is no communications network.
- The domain name server cannot be found.
- The domain name server does not recognize the machine.
- The command is run without super-user privileges.

Remote file sharing will not start if the host password in **/usr/nserve/loc.passwd** is corrupted. If you suspect this has happened, remove the file and run **rfstart** again to reenter your password.

Note: **rfstart** will *not* fail if your host password does not match the password on the domain name server. You will simply receive a warning message. However, if you try to mount a resource from the primary or any other host that validates your password, the mount will fail if your password does not match the one that host has listed for your machine.

OPTIONS

–v Specify that verification of all clients is required in response to initial incoming mount requests; any host not in the file **/usr/nserve/auth.info/***domain***/passwd** for the **domain** they belong to, will not be allowed to mount resources from your host. If the –v option is not specified, hosts named in *domain*/**passwd** will be verified, other hosts will be allowed to connect without verification.

–p *primary_addr*

Indicate the primary domain name server for your domain. *primary_addr* must be the network address of the primary name server for your domain. If the **–p** option is not specified, the address of the domain name server is taken from the **rfmaster** file. See **rfmaster**(5) for a description of the valid address syntax.

FILES

/usr/nserve/rfmaster
/usr/nserve/loc.passwd

SEE ALSO

rfmaster(5), **adv**(8), **dname**(8), **mount**(8), **rfadmin**(8), **rfstop**(8), **unadv**(8)

NAME

rfstop – stop the RFS environment

SYNOPSIS

rfstop

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rfstop disconnects a host from the Remote File Sharing (RFS) environment until another **rfstart**(8) is executed.

When executed on the domain name server, the domain name server responsibility is moved to a secondary name server as designated in the **rfmaster** file.

This command is restricted to the super-user.

If any of the following is true, an error message will be sent to standard error.

- There are resources currently advertised by this host.
- Resources from this machine are still remotely mounted by other hosts.
- There are still remotely mounted resources in the local file system tree.
- **rfstart**(8) had not previously been executed.
- The command is not run with super-user privileges.

SEE ALSO

rfmaster(5), **adv**(8), **mount**(8), **rfadmin**(8), **rfstart**(8), **unadv**(8)

NAME

rfuadmin – RFS notification shell script

SYNOPSIS

rfuadmin *message remote_resource* [*seconds*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

The **rfuadmin** shell script is used to respond to unexpected Remote File Sharing (RFS) events picked up by the **rfudaemon**(8) process. Such events may include broken network connections and forced unmounts. This script is not intended to be run directly from the shell.

Responses to messages received by **rfudaemon** can be tailored to suit the particular system by editing this script. The following paragraphs describe the arguments passed to **rfuadmin** and its standard responses.

disconnect *remote_resource*

A link to a remote resource has been cut. **rfudaemon** executes **rfuadmin**, passing it the message **disconnect** and the name of the disconnected resource. **rfuadmin** sends this message to all terminals using **wall**(1):

remote_resource **has been disconnected from the system.**

rfuadmin executes **fuser**(8) to kill all processes using the resource, unmounts the resource, and attempts to mount the resource again.

fumount remote_resource

A remote server machine has forced an unmount of a resource a local machine has mounted. The processing is similar to processing for a disconnect.

fuwarn *remote_resource seconds*

This message notifies **rfuadmin** that a resource is about to be unmounted. **rfudaemon** sends this script the **fuwarn** message, the resource name, and the number of seconds in which the forced unmount will occur. **rfuadmin** sends this message to all terminals:

remote_resource **is being removed from the system in # seconds.**

SEE ALSO

wall(1), **fumount**(8), **fuser**(8), **mount**(8), **rfstart**(8), **rfudaemon**(8)

BUGS

The console must be on when RFS is running, otherwise **rfuadmin** hangs when it attempts to write to it, in which case recovery from disconected resources may not complete.

NAME

rfudaemon – Remote File Sharing daemon

SYNOPSIS

rfudaemon

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

The RFS daemon, **rfudaemon**, is started automatically by **rfstart**(8) and runs as a daemon process while Remote File Sharing is active. It listens for unexpected events, such as broken network connections and forced unmounts, and invokes **rfuadmin**(8) to execute the appropriate administrative procedures. Events recognized by **rfudaemon** are as follows:

disconnect

A link to a remote resource has been cut. **rfudaemon** executes **rfuadmin**, with two arguments: **disconnect** and the name of the disconnected resource.

fumount

A remote server machine has forced an unmount of a resource a local machine has mounted. **rfudaemon** executes **rfuadmin**, with two arguments: **fumount** and the name of the disconnected resource.

getumsg

A remote user-level program has sent a message to the local **rfudaemon**. Currently the only message sent is **fuwarn**, which notifies **rfuadmin** that a resource is about to be unmounted. **rfudaemon** sends **rfuadmin** the **fuwarn**, the resource name, and the number of seconds in which the forced unmount will occur.

lastumsg

The local machine wants to stop the **rfudaemon** (**rfstop**(8)). This causes **rfudaemon** to exit.

SEE ALSO

rfstart(8), **rfstop**(8), **rfuadmin**(8)

NAME

rlogind, in.rlogind – remote login server

SYNOPSIS

/usr/etc/in.rlogind *host.port*

DESCRIPTION

rlogind is the server for the **rlogin**(1C) program. The server provides a remote login facility with authentication based on privileged port numbers.

rlogind is invoked by **inetd**(8C) when a remote login connection is established, and executes the following protocol:

- The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection. The client's address and port number are passed as arguments to **rlogind** by **inetd** in the form *host.port* with host in hex and port in decimal.
- The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see **hosts**(5)), the server aborts the connection.

Once the source port and address have been checked, **rlogind** allocates a pseudo-terminal (see **pty**(4)), and manipulates file descriptors so that the slave half of the pseudo-terminal becomes the **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of the **login**(1) program, invoked with the **–r** option. The login process then proceeds with the authentication process as described in **rshd**(8C), but if automatic authentication fails, it reprompts the user to login as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, the packet protocol described in **pty**(4) is invoked to provide ^S/^Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, **TERM**; see **environ**(5V).

SEE ALSO

inetd(8C)

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Try again.
A *fork* by the server failed.

/usr/bin/sh: ...
The user's login shell could not be started.

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

NAME

rmail – handle remote mail received via uucp

SYNOPSIS

rmail *recipient...*

DESCRIPTION

rmail interprets incoming mail received through **uucp**(1C), collapsing "From" lines in the form generated by **bin-mail (1)** (see **bin-mail**(1)) into a single line of the form *return-path*!*sender*, and passing the processed mail on to **sendmail**(8).

rmail is explicitly designed for use with **uucp**(1C) and **sendmail**(8).

SEE ALSO

bin-mail(1), **uucp**(1C), **sendmail**(8)

NAME

rm_client – remove an NFS client

SYNOPSIS

rm_client [**–y**] *clients*

DESCRIPTION

rm_client removes an NFS client from a server. By default, **rm_client** asks if you want to remove the client's root directory, swap file, hosts entry, and **/tftpboot** file and whether to delete the client's entry in **/etc/bootparams**. **rm_client** can be run only by the super-user on the server, while in multiuser mode, or while not in the miniroot.

OPTIONS

–y Supply "yes" answers to all questions about what to remove.

FILES

/etc/bootparams
/tftpboot/*machine_addr*
/export/root/*client*
/export/swap/*client*

SEE ALSO

add_client(8), **add_services**(8), **suninstall**(8)

Installing SunOS 4.1

DIAGNOSTICS

must be run as root (super-user).
You must be root to run **rm_client**.

NAME

rmntstat – display RFS mounted resource information

SYNOPSIS

rmntstat [**–h**] [*resource*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

When used with no options, **rmntstat** displays a list of all local Remote File Sharing resources that are remotely mounted, the local path name, and the corresponding clients. **rmntstat** returns the remote mount data regardless of whether a resource is currently advertised; this ensures that resources that have been unadvertised but are still remotely mounted are included in the report. When a *resource* is specified, **rmntstat** displays the remote mount information only for that resource.

This command is restricted to the super-user.

OPTIONS

–h Omit header information from the display.

EXIT STATUS

If no local resources are remotely mounted, **rmntstat** will return a successful exit status.

ERRORS

If *resource* does not physically reside on the local machine or is an invalid resource name, an error message will be sent to standard error.

SEE ALSO

mount(8), **fumount**(8), **unadv**(8)

NAME

rmt – remote magtape protocol module

SYNOPSIS

/usr/etc/rmt

DESCRIPTION

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. **rmt** is normally started up with an **rexec**(3N) or **rcmd**(3N) call.

The **rmt** program accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of

A*number*\n

where *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

E*error-number*\n*error-message*\n

where *error-number* is one of the possible error numbers described in **intro**(2) and *error-message* is the corresponding error string as printed from a call to **perror**(3). The protocol is comprised of the following commands:

S Return the status of the open device, as obtained with a **MTIOCGET ioctl** call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

C*device* Close the currently open device. The *device* specified is ignored.

I*operation*\n*count*\n
Perform a **MTIOCOP ioctl**(2) command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt_op* and *mt_count* fields of the structure used in the **ioctl** call. The return value is the *count* parameter when the operation is successful.

L*whence*\n*offset*\n
Perform an **lseek**(2V) operation using the specified parameters. The response value is that returned from the **lseek** call.

O*device*\n*mode*\n
Open the specified *device* using the indicated *mode*. *device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to **open**(2V). If a device had already been opened, it is closed before a new open is performed.

R*count* Read *count* bytes of data from the open device. **rmt** performs the requested **read**(2V) and responds with **A***count-read*\n if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.

W*count* Write data onto the open device. **rmt** reads *count* bytes from the connection, aborting if a premature EOF is encountered. The response value is that returned from the **write**(2V) call.

Any other command causes **rmt** to exit.

DIAGNOSTICS

All responses are of the form described above.

SEE ALSO

intro(2), **ioctl**(2), **lseek**(2V), **open**(2V), **read**(2V), **write**(2V), **perror**(3), **rcmd**(3N), **rexec**(3N), **mtio**(4), **dump**(8), **restore**(8)

BUGS

People tempted to use this for a remote file access protocol are discouraged.

NAME

route – manually manipulate the routing tables

SYNOPSIS

/usr/etc/route [**–fn**] **add** | **delete** [**host** | **net**] *destination* [*gateway* [*metric*]]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

route manually manipulates the network routing tables normally maintained by the system routing daemon, **routed**(8C), or through default routes and redirect messages from routers. **route** allows the superuser to operate directly on the routing table for the specific host or network indicated by *destination*. The *gateway* argument, if present, indicates the network gateway to which packets should be addressed. The *metric* argument indicates the number of "hops" to the *destination*. The *metric* is required for *add* commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways.

The **add** command instructs **route** to add a route to *destination*. **delete** deletes a route.

Routes to a particular host must be distinguished from those to a network. The optional keywords **net** and **host** force the destination to be interpreted as a network or a host, respectively. Otherwise, if the destination has a "local address part" of **INADDR_ANY**, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. If the route is to a destination connected by a gateway, the *metric* parameter should be greater than 0. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used directly for transmission. All symbolic names specified for a *destination* or *gateway* are looked up in the hosts database using **gethostbyname**( ) (see **gethostent**(3N)). If this lookup fails, then the name is looked up in the networks database using **getnetbyname**( ) (see **getnetent**(3N)). "default" is also a valid destination, which is used for all routes if there is no specific host or network route.

OPTIONS

–f Flush the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, **route** flushes the gateways before performing the command.

–n Prevents attempts to print host and network names symbolically when reporting actions. This is useful, for example, when all name servers are down on your local net, so you need a route before you can contact the name server.

FILES

/etc/hosts
/etc/networks

SEE ALSO

ioctl(2), **gethostent**(3N), **getnetent**(3N), **routing**(4N), **routed**(8C)

DIAGNOSTICS

add [**host** | **net**] *destination*:*gateway*
The specified route is being added to the tables. The values printed are from the routing table entry supplied in the **ioctl**(2) call.

delete [**host** | **net**] *destination*:*gateway*
The specified route is being deleted.

*destination***done**
When the **–f** flag is specified, each routing table entry deleted is indicated with a message of this form.

Network is unreachable

An attempt to add a route failed because the gateway listed was not on a directly-connected network. Give the next-hop gateway instead.

not in table

A delete operation was attempted for an entry that is not in the table.

routing table overflow

An add operation was attempted, but the system was unable to allocate memory to create the new entry.

NAME

routed, in.routed – network routing daemon

SYNOPSIS

/usr/etc/in.routed [**–qstv**] [*logfile*]

DESCRIPTION

routed is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries.

In normal operation **routed** listens on **udp**(4P) socket 520 (decimal) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When **routed** is started, it uses the **SIOCGIFCONF ioctl( )** (see **ioctl**(2)) to find those directly connected interfaces configured into the system and marked "up" (the software loopback interface is ignored). If multiple interfaces are present, it is assumed the host will forward packets between networks. **routed** then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, **routed** formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned provides a metric *relative to the sender*.

request packets received by **routed** are used to update the routing tables if one of the following conditions is satisfied:

- No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (that is, the hop count is not infinite).
- The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.
- The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, **routed** records the change in its internal tables and generates a *response* packet to all directly connected hosts and networks. **routed** waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables to allow possible unstable situations to settle.

In addition to processing incoming packets, **routed** also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks.

In addition to the facilities described above, **routed** supports the notion of "distant" *passive* and *active* gateways. When **routed** is started up, it reads the file **/etc/gateways** to find gateways which may not be identified using the **SIOGIFCONF ioctl( )**. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (that is, they should have a **routed** process running on the machine). Passive gateways are maintained in the routing tables forever and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted.

The **/etc/gateways** is comprised of a series of lines, each in the following format:

< net | host > *filename1* **gateway** *filename2* **metric** *value* **< passive | active >**

The **net** or **host** keyword indicates if the route is to a network or specific host.

filename1 is the name of the destination network or host. This may be a symbolic name located in **/etc/networks** or **/etc/hosts**, or an Internet address specified in "dot" notation; see **inet**(3N).

filename2 is the name or address of the gateway to which messages should be forwarded.

value is a metric indicating the hop count to the destination host or network.

The keyword **passive** or **active** indicates if the gateway should be treated as passive or active (as described above).

OPTIONS

–s Force **routed** to supply routing information whether it is acting as an internetwork router or not.

–q Opposite of the **–s** option.

–t All packets sent or received are printed on the standard output. In addition, **routed** will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process.

–v Allow a logfile to be created showing the changes made to the routing tables with a timestamp.

logfile Specify a file in which **routed** records any changes to the routing tables and a history of recent messages sent and received which are related to the changed route.

FILES

/etc/gateways for distant gateways
/etc/networks
/etc/hosts

SEE ALSO

ioctl(2), **inet**(3N), **udp**(4P)

BUGS

The kernel's routing tables may not correspond to those of **routed** for short periods of time while processes utilizing existing routes exit; the only remedy for this is to place the routing process in the kernel.

routed should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information.

NAME

rpcinfo – report RPC information

SYNOPSIS

rpcinfo **–p** [*host*]

rpcinfo [**–n** *portnum*] **–u** *host program* [*version*]

rpcinfo [**–n** *portnum*] **–t** *host program* [*version*]

rpcinfo **–b** *program version*

rpcinfo **–d** *program version*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rpcinfo makes an RPC call to an RPC server and reports what it finds.

OPTIONS

–p Probe the portmapper on *host*, and print a list of all registered RPC programs. If *host* is not specified, it defaults to the value returned by **hostname**(1).

–u Make an RPC call to procedure 0 of *program* on the specified *host* using UDP, and report whether a response was received.

–t Make an RPC call to procedure 0 of *program* on the specified *host* using TCP, and report whether a response was received.

–n Use *portnum* as the port number for the *–t* and *–u* options instead of the port number given by the portmapper.

–b Make an RPC broadcast to procedure 0 of the specified *program* and *version* using UDP and report all hosts that respond.

–d Delete registration for the RPC service of the specified *program* and *version*. This option can be exercised only by the super-user.

The *program* argument can be either a name or a number.

If a *version* is specified, **rpcinfo** attempts to call that version of the specified *program*. Otherwise, **rpcinfo** attempts to find all the registered version numbers for the specified *program* by calling version 0 (which is presumed not to exist; if it does exist, **rpcinfo** attempts to obtain this information by calling an extremely high version number instead) and attempts to call each registered version. Note: the version number is required for **–b** and **–d** options.

EXAMPLES

To show all of the RPC services registered on the local machine use:

example% rpcinfo –p

To show all of the RPC services registered on the machine named **klaxon** use:

example% rpcinfo –p klaxon

To show all machines on the local net that are running the Network Interface Service (NIS) use:

example% rpcinfo –b ypserv 'version' | uniq

where 'version' is the current NIS version obtained from the results of the **–p** switch above.

To delete the registration for version 1 of the **walld** service use:

example% rpcinfo –d walld 1

SEE ALSO

rpc(5), **portmap**(8C)

RPC Programming Guide in *Network Programming*

BUGS

In releases prior to the SunOS 3.0 release, the Network File System (NFS) did not register itself with the portmapper; **rpcinfo** cannot be used to make RPC calls to the NFS server on hosts running such releases.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

rquotad, rpc.rquotad – remote quota server

SYNOPSIS

/usr/etc/rpc.rquotad

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rquotad is an **rpc**(3N) server which returns quotas for a user of a local file system which is mounted by a remote machine over the NFS. The results are used by **quota**(1) to display user quotas for remote file systems. The **rquotad** daemon is normally invoked by **inetd**(8C).

FILES

quotas	quota file at the file system root

SEE ALSO

quota(1), **rpc**(3N), **nfs**(4P), **services**(5) **inetd**(8C)

NAME

rshd, in.rshd – remote shell server

SYNOPSIS

/usr/etc/in.rshd *host.port*

DESCRIPTION

rshd is the server for the **rcmd**(3N) routine and, consequently, for the **rsh**(1C) program. The server provides remote execution facilities with authentication based on privileged port numbers.

rshd is invoked by **inetd**(8C) each time a shell service is requested, and executes the following protocol:

- The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection. The clients host address (in hex) and port number (in decimal) are the argument passed to **rshd.**
- The server reads characters from the socket up to a null (**\0**) byte. The resultant string is interpreted as an ASCII number, base 10.
- If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 512-1023.
- The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see **hosts**(5)), the server aborts the connection.
- A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the **server**'s machine.
- A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the **client**'s machine.
- A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
- **rshd** then validates the user according to the following steps. The remote user name is looked up in the password file and a **chdir** is performed to the user's home directory. If the lookup or fails, the connection is terminated. If the **chdir** fails, it does a **chdir** to / **(root).** If the user is not the super-user, (user ID 0), the file **/etc/hosts.equiv** is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file **.rhosts** in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated.
- A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rshd.**

FILES

/etc/hosts.equiv

SEE ALSO

rsh(1C), **rcmd**(3N), **syslogd**(8)

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

DIAGNOSTICS

The following diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the command execution).

locuser too long
The name of the user on the client's machine is longer than 16 characters.

remuser too long
The name of the user on the remote machine is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Login incorrect.
No password file entry for the user name existed.

Permission denied.
The authentication procedure described above failed.

Can't make pipe.
The pipe needed for the **stderr**, was not created.

Try again.
A *fork* by the server failed.

/usr/bin/sh: ...
The user's login shell could not be started.

In addition, daemon's status messages and internal diagnostics are logged to the appropriate system log using the **syslogd**(8) facility.

NAME

rstatd, rpc.rstatd – kernel statistics server

SYNOPSIS

/usr/etc/rpc.rstatd

DESCRIPTION

rstatd is a server which returns performance statistics obtained from the kernel. These statistics are graphically displayed by **perfmeter**(1). The **rstatd** daemon is normally invoked by **inetd**(8C).

Systems with disk drivers to be monitored by this daemon must be configured so as to report disk (**_dk_xfer**) statistics.

SEE ALSO

perfmeter(1), **services**(5), **inetd**(8C)

NAME

runacct – run daily accounting

SYNOPSIS

/usr/lib/acct/runacct [*mmdd* [*state*]]

DESCRIPTION

runacct is the main daily accounting shell procedure. It is normally initiated using **cron**(8). **runacct** processes connect, fee, disk, and process accounting files. It also prepares summary files for **prdaily** or billing purposes.

runacct takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into **active**. When an error is detected, a message is written to **/dev/console**, mail (see **mail**(1)) is sent to **root**, and **runacct** terminates. **runacct** uses a series of lock files to protect against re-invocation. The files **lock** and **lock1** are used to prevent simultaneous invocation, and **lastdate** is used to prevent more than one invocation per day.

runacct breaks its processing into separate, restartable *states* using **statefile** to remember the last *state* completed. It accomplishes this by writing the *state* name into **statefile**. **runacct** then looks in **statefile** to see what it has done and to determine what to process next. *states* are executed in the following order:

SETUP	Move active accounting files into working files.
WTMPFIX	Verify integrity of the **wtmp** file, correcting date changes if necessary.
CONNECT1	Produce connect session records in **ctmp.h** format.
CONNECT2	Convert **ctmp.h** records into **tacct.h** format.
PROCESS	Convert process accounting records into **tacct.h** format.
MERGE	Merge the connect and process accounting records.
FEES	Convert output of **chargefee** into **tacct.h** format and merge with connect and process accounting records.
DISK	Merge disk accounting records with connect, process, and fee accounting records.
MERGETACCT	Merge the daily total accounting records in **daytacct** with the summary total accounting records in **/var/adm/acct/sum/tacct**.
CMS	Produce command summaries.
USEREXIT	Any installation-dependent accounting programs can be included here.
CLEANUP	Cleanup temporary files and exit.

To restart **runacct** after a failure, first check the **active** file for diagnostics, then fix up any corrupted data files, such as **pacct** or **wtmp**. The **lock** files and **lastdate** file must be removed before **runacct** can be restarted. The argument *mmdd* is necessary if **runacct** is being restarted, and specifies the month and day for which **runacct** will rerun the accounting. Entry point for processing is based on the contents of **statefile**; to override this, include the desired *state* on the command line to designate where processing should begin.

EXAMPLES

To start **runacct**:

nohup runacct 2> /var/adm/acct/nite/fd2log &

To restart **runacct**:

nohup runacct 0601 2>> /var/adm/acct/nite/fd2log &

To restart **runacct** at a specific *state*:

nohup runacct 0601 MERGE 2>> /var/adm/acct/nite/fd2log &

FILES

/etc/wtmp
/var/adm/pacct*
/var/adm/acct/nite/active
/var/adm/acct/nite/daytacct
/var/adm/acct/nite/lock
/var/adm/acct/nite/lock1
/var/adm/acct/nite/lastdate
/var/adm/acct/nite/statefile
/var/adm/acct/nite/ptacct*.*mmdd*

SEE ALSO

acctcom(1), **mail**(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **cron**(8), **fwtmp**(8)

BUGS

Normally it is not a good idea to restart **runacct** in the **SETUP** *state*. Run **SETUP** manually and restart using:

runacct *mmdd* **WTMPFIX**

If **runacct** failed in the **PROCESS** *state*, remove the last **ptacct** file because it will not be complete.

NAME

rusage – print resource usage for a command

SYNOPSIS

rusage *command*

DESCRIPTION

The **rusage** command is similar to **time**(1V). It runs the given *command*, which must be specified; that is, *command* is not optional as it is in the C shell's timing facility. When the command is complete, **rusage** displays the real (wall clock), the system CPU, and the user CPU times which elapsed during execution of the command, plus other fields in the **rusage** structure, all on one long line. Times are reported in seconds and hundredths of a second.

EXAMPLE

The example below shows the format of **rusage** output.

```
example% rusage wc /usr/man/man1/csh (1)
3045  13423  78071 /usr/man/man1/csh (1)
2.26 real 0.80 user 0.36 sys 11 pf 38 pr 0 sw 11 rb 0 wb 16 vcx 37 icx 24 mx 0 ix 1230 id 9 is
example%
```

Each of the fields identified corresponds to an element of the **rusage** structure, as described in **getrusage**(2), as follows:

real		**elapsed real time**
user	**ru_utime**	**user time used**
sys	**ru_stime**	**system time used**
pf	**ru_majflt**	**page faults requiring physical I/O**
pr	**ru_minflt**	**page faults not requiring physical I/O**
sw	**ru_nswap**	**swaps**
rb	**ru_inblock**	**block input operations**
wb	**ru_oublock**	**block output operations**
vcx	**ru_nvcsw**	**voluntary context switches**
icx	**ru_nivcsw**	**involuntary context switches**
mx	**ru_maxrss**	**maximum resident set size**
ix	**ru_ixrss**	**currently 0**
id	**ru_idrss**	**integral resident set size**
is	**ru_isrss**	**currently 0**

SEE ALSO

csh(1), **time**(1V), **getrusage**(2)

BUGS

When the command being timed is interrupted, the timing values displayed may be inaccurate.

NAME

rusersd, rpc.rusersd – network username server

SYNOPSIS

/usr/etc/rpc.rusersd

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rusersd is a server that returns a list of users on the network. The **rusersd** daemon is normally invoked by **inetd**(8C).

SEE ALSO

perfmeter(1), **rusers**(1C), **services**(5) **inetd**(8C)

Installing SunOS 4.1

NAME

rwalld, rpc.rwalld – network rwall server

SYNOPSIS

/usr/etc/rpc.rwalld

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rwalld is a server that handles **rwall**(1C) and **shutdown**(2) requests. It is implemented by calling **wall**(1) to all the appropriate network machines. The **rwalld** daemon is normally invoked by **inetd**(8C).

SEE ALSO

rwall(1C), **wall**(1), **shutdown**(2) **services**(5), **inetd**(8C)

NAME

rwhod, in.rwhod – system status server

SYNOPSIS

/usr/etc/in.rwhod

AVAILABILITY

Due to its potential impact on network performance, this service is commented out of the **/etc/rc** system initialization script. It is provided only for 4.3 BSD compatibility.

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rwhod is the server which maintains the database used by the **rwho**(1C) and **ruptime**(1C) programs. Its operation is predicated on the ability to *broadcast* messages on a network.

rwhod operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast on a network. As a consumer of information, it listens for other **rwhod** servers' status messages, validating them, then recording them in a collection of files located in the directory **/var/spool/rwho**.

The **rwho** server transmits and receives messages at the port indicated in the "rwho" service specification, see **services**(5). The messages sent and received, are of the form:

```
struct   outmp {
         char    out_line[8];      /* tty name */
         char    out_name[8];      /* user id */
         long    out_time;         /* time on */
};

struct   whod {
         char    wd_vers;
         char    wd_type;
         char    wd_fill[2];
         int     wd_sendtime;
         int     wd_recvtime;
         char    wd_hostname[32];
         int     wd_loadav[3];
         int     wd_boottime;
                 struct   whoent {
                 struct   outmp we_utmp;
                 int      we_idle;
         } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the **w**(1) program, and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the **gethostname**(2) system call. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the **utmp**(5V) entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the **rwho** server are discarded unless they originated at a **rwho** server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by **rwhod** are placed in files named **whod.***hostname* in the directory **/var/spool/rwho**. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 60 seconds. **rwhod** performs an **nlist (3V)** on **/vmunix** every 10 minutes to guard against the possibility that this file is not the system image currently operating.

FILES

/etc/rc
/var/spool/rwho

SEE ALSO

rwho(1C), **ruptime**(1C), **w**(1), **gethostname**(2), **nlist**(3V), **utmp**(5V), **syslogd**(8)

DIAGNOSTICS

Status and diagnostic messages are logged to the appropriate system log using the **syslogd**(8) facility.

BUGS

This service takes up progressively more network bandwidth as the number of hosts on the local net increases. For large networks, the cost becomes prohibitive. RPC-based services such as **rup**(1C) and **rusers**(1C) provide a similar function with greater efficiency.

rwhod should relay status information between networks. People often interpret the server dying as a machine going down.

NAME

sa, accton – system accounting

SYNOPSIS

/usr/etc/sa [**–abcdDfijkKlmnrstu**] [–**v**[*n*]] [–**S** *savacctfile*] [–**U** *usracctfile*] [*filename*]

/usr/lib/acct/accton [*filename*]

DESCRIPTION

With an argument naming an existing *filename*, **accton** causes system accounting information for every process executed to be placed at the end of the file. If no argument is given, accounting is turned off.

sa reports on, cleans up, and generally maintains accounting files.

sa is able to condense the information in **/var/adm/pacct** into a summary file **/var/adm/savacct** which contains a count of the number of times each command was called and the time resources consumed. This condensation is desirable because on a large system **/var/adm/pacct** can grow by 500K bytes per day. The summary file is normally read before the accounting file, so the reports include all available information.

If a file name is given as the last argument, that file will be treated as the accounting file; **/var/adm/pacct** is the default.

Output fields are labeled: **cpu** for the sum of user+system time (in minutes), **re** for real time (also in minutes), **k** for CPU-time averaged core usage (in 1k units), **avio** for average number of I/O operations per execution. With options fields labeled **tio** for total I/O operations, **k*sec** for CPU storage integral (kilo-core seconds), **u** and **s** for user and system CPU time alone (both in minutes) will sometimes appear.

sa also breaks out accounting statistics by user. This information is kept in the file **/var/adm/usracct**.

OPTIONS

–a Print all command names, even those containing unprintable characters and those used only once. By default, those are placed under the name '***other.'

–b Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system times.

–c Besides total user, system, and real time for each command print percentage of total time over all commands.

–d Sort by average number of disk I/O operations.

–D Print and sort by total number of disk I/O operations.

–f Force no interactive threshold compression with **–v** flag.

–i Do not read in summary file.

–j Instead of total minutes time for each category, give seconds per call.

–k Sort by CPU-time average memory usage.

–K Print and sort by CPU-storage integral.

–l Separate system and user time; normally they are combined.

–m Print number of processes and number of CPU minutes for each user.

–n Sort by number of calls.

–r Reverse order of sort.

–s Merge accounting file into summary file **/var/adm/savacct** when done.

–t For each command report ratio of real time to the sum of user and system times.

–u Superseding all other flags, print for each record in the accounting file the user ID and command name.

–v Followed by a number *n*, types the name of each command used *n* times or fewer. If *n* is not specified, it defaults to 1. Await a reply from the terminal; if it begins with **y**, add the command to the category '**junk**.' This is used to strip out garbage.

–S The following filename is used as the command summary file instead of **/var/adm/savacct**.

–U The following filename is used instead of **/var/adm/usracct** to accumulate the per-user statistics printed by the **–m** option.

FILES

/var/adm/pacct	raw accounting
/var/adm/savacct	summary by command
/var/adm/usracct	summary by user ID

SEE ALSO

acct(2V), **acct**(5), **ac**(8)

BUGS

sa's execution time increases linearly with the magnitude of the largest positive user ID in **/etc/passwd**.

NAME

savecore – save a core dump of the operating system

SYNOPSIS

/usr/etc/savecore [**–v**] *directory* [*system-name*]

DESCRIPTION

savecore saves a core dump of the kernel (assuming that one was made) and writes a reboot message in the shutdown log. It is meant to be called near the end of the **/etc/rc.local** file after the system boots. However, it is not normally run by default. You must edit that file to enable it.

savecore checks the core dump to be certain it corresponds with the version of the operating system currently running. If it does, **savecore** saves the core image in the file *directory*/**vmcore.***n* and the kernel's namelist in *directory*/**vmunix.***n*. The trailing *.n* in the pathnames is replaced by a number which grows every time **savecore** is run in that directory.

Before **savecore** writes out a core image, it reads a number from the file *directory*/**minfree**. This is the minimum number of kilobytes that must remain free on the filesystem containing *directory*. If there is less free space on the filesystem containing *directory* than the number of kilobytes specified in **minfree**, the core dump is not saved. If the **minfree** file does not exist, **savecore** always writes out the core file (assuming that a core dump was taken).

savecore also logs a reboot message using facility **LOG_AUTH** (see **syslog**(3)). If the system crashed as a result of a panic, **savecore** logs the panic string too.

If the core dump was from a system other than **/vmunix**, the name of that system must be supplied as *system-name*.

OPTIONS

–v Verbose. Enable verbose error messages from **savecore**.

FILES

directory/**vmcore.***n*
directory/**vmunix.***n*
directory/**minfree**
/vmunix the kernel
/etc/rc.local

SEE ALSO

syslog(3), **panic**(8S), **sa**(8)

BUGS

savecore can be fooled into thinking a core dump is the wrong size.

You must run **savecore** very soon after booting — before the swap space containing the crash dump is overwritten by programs currently running.

NAME

sendmail – send mail over the internet

SYNOPSIS

/usr/lib/sendmail [**–ba**] [**–bd**] [**–bi**] [**–bm**] [**–bp**] [**–bs**] [**–bt**] [**–bv**] [**–bz**]
[**–C***file*] [**–d***X*] [**–F***fullname*] [**–f***name*] [**–h***N*] [**–n**] [**–o***x value*] [**–q**[*time*]]
[**–r***name*] [**–R***string*] [**–t**] [**–v**] [*address* ...]

DESCRIPTION

sendmail sends a message to one or more people, routing the message over whatever networks are necessary. **sendmail** does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; **sendmail** is used only to deliver pre-formatted messages.

With no flags, **sendmail** reads its standard input up to an EOF, or a line with a single dot and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in the local **aliases**(5) file, or by using the Network Interface Service (NIS), and aliased appropriately. In addition, if there is a **.forward** file in a recipient's home directory, **sendmail** forwards a copy of each message to the list of recipients that file contains. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in alias expansions, for example, if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

sendmail will also route mail directly to other known hosts in a local network. The list of hosts to which mail is directly sent is maintained in the file **/usr/lib/mailhosts**.

OPTIONS

–ba Go into ARPANET mode. All input lines must end with a LINEFEED, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.

–bd Run as a daemon, waiting for incoming SMTP connections.

–bi Initialize the alias database.

–bm Deliver mail in the usual way (default).

–bp Print a summary of the mail queue.

–bs Use the SMTP protocol as described in RFC 821. This flag implies all the operations of the **–ba** flag that are compatible with SMTP.

–bt Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

–bv Verify names only — do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

–bz Create the configuration freeze file.

–C*file* Use alternate configuration file.

–d*X* Set debugging value to *X*.

–F*fullname* Set the full name of the sender.

–f*name* Sets the name of the "from" person (that is, the sender of the mail). **–f** can only be used by "trusted" users (who are listed in the config file).

–h*N* Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.

–M*id* Attempt to deliver the queued message with message-id **id**.

–n Do not do aliasing.

–o*x value* Set option *x* to the specified *value*. Options are described below.

–q[*time*] Processed saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *time* is given as a tagged number, with **s** being seconds, **m** being minutes, **h** being hours, **d** being days, and **w** being weeks. For example, **–q1h30m** or **–q90m** would both set the timeout to one hour thirty minutes.

–r*name* An alternate and obsolete form of the **–f** flag.

–R*string* Go through the queue of pending mail and attempt to deliver any message with a recipient containing the specified string. This is useful for clearing out mail directed to a machine which has been down for awhile.

–t Read message for recipients. "To:", "Cc:", and "Bcc:" lines will be scanned for people to send to. The "Bcc:" line will be deleted before transmission. Any addresses in the argument list will be suppressed.

–v Go into verbose mode. Alias expansions will be announced, etc.

PROCESSING OPTIONS

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the **–o** flag or in the configuration file. These are described in detail in the *Installation and Operation Guide*. The options are:

A*file* Use alternate alias file.

c On mailers that are considered "expensive" to connect to, do not initiate immediate connection. This requires queueing.

d*x* Set the delivery mode to **x**. Delivery modes are **i** for interactive (synchronous) delivery, **b** for background (asynchronous) delivery, and **q** for queue only — that is, actual delivery is done the next time the queue is run.

D Run **newaliases**(8) to automatically rebuild the alias database, if necessary.

e*x* Set error processing to mode **x**. Valid modes are **m** to mail back the error message, **w** to "write" back the error message (or mail it back if the sender is not logged in), **p** to print the errors on the terminal (default), 'q' to throw away error messages (only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by modes **m** or **w** and if the sender is local to this machine, a copy of the message is appended to the file **dead.letter** in the sender's home directory.

F*mode* The mode to use when creating temporary files.

f Save UNIX-system-style "From" lines at the front of messages.

g*N* The default group ID to use when calling mailers.

H*file* The SMTP help file.

i Do not take dots on a line by themselves as a message terminator.

L*n* The log level.

m Send to "me" (the sender) also if I am in an alias expansion.

o If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (that is, commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

Q*queuedir*
Select the directory in which to queue messages.

r*timeout*
The timeout on reads; if none is set, **sendmail** will wait forever for a mailer.

S*file* Save statistics in the named file.

s Always instantiate the queue file, even under circumstances where it is not strictly necessary.

T*time* Set the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.

t*stz,dtz* Set the name of the time zone.

u*N* Set the default user id for mailers.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep **sendmail** from suppressing the blanks from between arguments.

sendmail returns an exit status describing what it did. The codes are defined in **sysexits.h**

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as "cannot fork".
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as **newaliases**, **sendmail** rebuilds the alias database. If invoked as **mailq**, **sendmail** prints the contents of the mail queue.

FILES

Except for **/etc/sendmail.cf**, these pathnames are all specified in **/etc/sendmail.cf**. Thus, these values are only approximations.

/etc/aliases	raw data for alias names
/etc/aliases.pag	data base of alias names
/etc/aliases.dir	
/usr/lib/mailhosts	list of hosts to which mail can be sent directly
/etc/sendmail.cf	configuration file
/etc/sendmail.fc	frozen configuration
/etc/sendmail.hf	help file
/etc/sendmail.st	collected statistics
/usr/bin/uux	to deliver uucp mail
/usr/bin/mail	to deliver local mail
/var/spool/mqueue/*	temp files and queued mail
~/.forward	list of recipients for forwarding messages

SEE ALSO

biff(1), **bin-mail**(1), **mail**(1), **aliases**(5) **newaliases**(8)

System and Network Administration

Su, Zaw-Sing, and Jon Postel, *The Domain Naming Convention for Internet User Applications*, RFC 819, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Postel, Jon, *Simple Mail Transfer Protocol*, RFC 821, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Crocker, Dave, *Standard for the Format of ARPA-Internet Text Messages*, RFC 822, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

set4, unset4, check4 – set, unset, and check the 4 megabyte process virtual address space limit flag in a Sun386i module

SYNOPSIS

set4 [**–d** *working_directory*] [– | *filename*] ...

unset4 *filename* ...

check4 *filename* ...

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

set4 sets the 4 megabyte process memory flag in each *filename* program image, limiting the virtual address space for each program to 4 megabytes. If a '–' is used, **set4** reads the standard input for a list of files to set the 4 megabyte limit on. Lines in the standard input whose first character is '#' are ignored, so files may include comments.

unset4 clears the 4 megabyte process memory flag in the program image, so the process virtual address space is not limited to 4 megabytes.

check4 reports programs that do not have the 4 megabyte limit set, and does not report programs with the limit set.

OPTIONS

–d *working_directory*

This specifies a directory prefix for file names that **set4** processes.

EXAMPLES

Suppose that the file **small_progs** contains the following:

```
# These files should have their virtual address spaces limited to 4 MB:
/bin/date
/bin/true
```

Then the following command will run **set4** on **/build/bin/false**, **/build/bin/date**, **/build/bin/true**, and **/build/bin/cat.**

```
example% set4 –d /build /bin/false –
/bin/cat < small_progs
example%
```

In this example, **unset4** clears the 4 megabyte limit flag in **date**, and **clri**.

```
example% unset4 /bin/date /etc/clri
example%
```

In the last example, **check4** shows that **date** and **clri** are 4 megabyte processes, but **basename** is not.

```
example% check4 /bin/date /etc/clri /usr/bin/basename
basename is not a 4MB process
example%
```

SEE ALSO

execve(2V) **execl**(3V)

BUGS

There is a problem in the way that processes that have the 4 megabyte limit set **exec**( ) processes that do not have the limit set. (See **execve**(2V) and **execl**(3V) for descriptions of **exec**( ) processing.) For a short time during the **exec**( ), a child has the parent's data and stack limits. During this time, the program is checked

to see if it will fit into memory. If the parent had the 4 megabyte limit set, the test fails, because the child program is running with the parent's 4 megabyte limit. This only affects programs which have more than 4 megabytes of global or static data compiled into the program. It does not affect programs which use **malloc**(3V) to obtain memory.

For example, **csh**(1) and **sh**(1) may be 4 megabyte processes. If they are, and if you try to run a program with more than 4 megabytes of global and static data, the shell cannot successfully **exec**(). To fix this problem, become root on your machine and enter the following commands:

```
example% /etc/mount -o remount,rw /usr
/usr/etc/unset4 /bin/csh /bin/sh
example%
```

Then log out and back in again to run the modified shell. This makes **csh** and **sh** "normal" processes.

NAME

setsid – set process to session leader

SYNOPSIS

setsid [**–b**] *command* [*arguments*]

DESCRIPTION

setsid executes *command* after altering the execution environment such that the next non-controlling terminal opened will be assigned as *command*'s controlling terminal.

OPTIONS

–b Alteration to the execution environment persists across calls to **fork**(2V).

The **–b** option puts the process into a state that is supported in SunOS Release 4.1 solely as a migration aid; this option will not be supported in future releases.

EXAMPLES

Components of two SunLink products, **/usr/sunlink/dni/dnilogind** (the DECNET analog of **rlogind**(8C) and **/usr/sunlink/x25/x29** (the OSI analog of **rlogind**), are known to need this wrapper. Typical usage is:

```
example% cd /usr/sunlink/dni
example% mv dnilogind .dnilogind
example% cat > dnilogind
#!/bin/sh
/usr/etc/setsid –b /usr/sunlink/dni/.dnilogind "$@"
^D
example% chmod +x dnilogind
```

SEE ALSO

setsid(2V)

IEEE Std 1003.1-1988

NAME

showfh – print full pathname of file from the NFS file handle

SYNOPSIS

/usr/etc/showfh *server_name num1 num2 ... num8*

DESCRIPTION

showfh prints the full path name of the file on the server for the given file handle (*num1 ... num8*). *server_name* is the server from where the client got this file handle. *num1 ... num8* are the file handle numbers represented in hexadecimal notation.

The **showfhd** daemon should be running on the NFS servers to answer **showfh** requests. If it cannot find the file corresponding to the given file handle, it prints a diagnostic message.

SEE ALSO

showfhd(8C)

BUGS

If the given NFS file handle is stale, then **showfh** may not print the name of the actual file. The inode for the file could have been allocated to some other file.

NAME

showfhd – showfh daemon run on the NFS servers

SYNOPSIS

/usr/etc/rpc.showfhd

DESCRIPTION

showfhd is the daemon which runs on the NFS servers and answers **showfh** requests. It provides the full path name for the given file handle. If it cannot find the file for the corresponding inode number, it returns an error message.

FILES

/etc/mtab	table of mounted file systems

SEE ALSO

find(1), **showfh**(8C)

NAME

showmount – show all remote mounts

SYNOPSIS

/usr/etc/showmount [**–ade**] [*hostname*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

showmount lists all the clients that have remotely mounted a filesystem from *host*. This information is maintained by the **mountd**(8C) server on *host*, and is saved across crashes in the file **/etc/rmtab**. The default value for *host* is the value returned by **hostname**(1).

OPTIONS

–a Print all remote mounts in the format:

hostname:*directory*

where *hostname* is the name of the client, and **directory** is the root of the file system that has been mounted.

–d List directories that have been remotely mounted by clients.

–e Print the list of exported file systems.

FILES

/etc/rmtab

SEE ALSO

hostname(1), **exports**(5), **exports**(5), **exportfs**(8), **mountd**(8C)

BUGS

If a client crashes, its entry will not be removed from the list until it reboots and executes '**umount –a**'.

NAME

shutdown – close down the system at a given time

SYNOPSIS

/usr/etc/shutdown [**–fhknr**] [*time* [*warning-message* ...]

DESCRIPTION

shutdown provides an automated procedure to notify users when the system is to be shut down. *time* specifies when **shutdown** will bring the system down; it may be the word **now** (indicating an immediate shutdown), or it may specify a future time in one of two formats: *+number* and *hour:min*. The first form brings the system down in *number* minutes, and the second brings the system down at the time of day indicated in 24-hour notation.

At intervals that get closer as the apocalypse approaches, warning messages are displayed at terminals of all logged-in users, and of users who have remote mounts on that machine. Five minutes before shutdown, or immediately if shutdown is in less than 5 minutes, logins are disabled by creating **/etc/nologin** and writing a message there. If this file exists when a user attempts to log in, **login**(1) prints its contents and exits. The file is removed just before **shutdown** exits.

At shutdown time a message is written to the system log daemon, **syslogd**(8), containing the time of shutdown, the instigator of the shutdown, and the reason. Then a terminate signal is sent to **init**, which brings the system down to single-user mode.

The time of the shutdown and the warning message are placed in **/etc/nologin**, which should be used to inform the users as to when the system will be back up, and why it is going down (or anything else).

OPTIONS

As an alternative to the above procedure, these options can be specified:

–f Shut the system down in the manner of **fasthalt** (see **fastboot**(8)), so that when the system is rebooted, the file systems are not checked.

–h Execute **halt**(8).

–k Simulate shutdown of the system. Do not actually shut down the system.

–n Prevent the normal **sync**(2) before stopping.

–r Execute **reboot**(8).

FILES

/etc/nologin	tells login not to let anyone log in
/etc/xtab	list of remote hosts that have mounted this host

SEE ALSO

login(1), **sync**(2), **fastboot**(8), **halt**(8), **reboot**(8), **syslogd**(8)

BUGS

Only allows you to bring the system down between "now" and 23:59 if you use the absolute time for shutdown.

NAME

skyversion – print the SKYFFP board microcode version number

SYNOPSIS

/usr/etc/skyversion

DESCRIPTION

skyversion obtains from the SKYFFP board the Sky version number of the microcode currently loaded and prints the result on the standard output.

DIAGNOSTICS

The Sky version number operation code used to implement this command is not available for microcode releases earlier than Sky release 3.00. The result in this case is unpredictable and is either a nonmeaningful version number or a message indicating that no version number is available.

Meaningful version numbers are of the form *n.dd* where $n \geq 3$.

NAME

spray – spray packets

SYNOPSIS

/usr/etc/spray [**–c** *count*] [**–d** *delay*] [**–i** *delay*] [**–l** *length*] *host*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

spray sends a one-way stream of packets to *host* using RPC, and reports how many were received, as well as the the transfer rate. The *host* argument can be either a name or an internet address.

OPTIONS

–c *count* Specify how many packets to send. The default value of *count* is the number of packets required to make the total stream size 100000 bytes.

–d *delay* Specify how many microseconds to pause between sending each packet. The default is 0.

–i *delay* Use ICMP echo packets rather than RPC. Since ICMP automatically echos, this creates a two way stream.

–l *length* The *length* parameter is the numbers of bytes in the Ethernet packet that holds the RPC call message. Since the data is encoded using XDR, and XDR only deals with 32 bit quantities, not all values of *length* are possible, and **spray** rounds up to the nearest possible value. When *length* is greater than 1514, then the RPC call can no longer be encapsulated in one Ethernet packet, so the *length* field no longer has a simple correspondence to Ethernet packet size. The default value of *length* is 86 bytes (the size of the RPC and UDP headers).

SEE ALSO

icmp(4P), **ping**(8C), **sprayd**(8C)

Installing SunOS 4.1

NAME

sprayd, rpc.sprayd – spray server

SYNOPSIS

/usr/etc/rpc.sprayd

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rpc.sprayd is a server which records the packets sent by **spray**(8C), and sends a response to the originator of the packets. The **rpc.sprayd** daemon is normally invoked by **inetd**(8C).

SEE ALSO

inetd(8C), **spray**(8C)

Installing SunOS 4.1

NAME

start_applic – generic application startup procedures

SYNOPSIS

/usr/etc/start_applic

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

start_applic is a short generic shell script that can be copied or symbolically linked into either **/vol/local/bin/***application* or **/usr/local/bin/***application*. When invoked as *application*, an application installed as described below will be correctly invoked on systems of any supported processor architecture. Installing **start_applic** (or a customized version of it) in one of these locations ensures that no user's or system's environment needs to be modified just to run the application. Applications are stored in a single tree which is not shared with any other applications. This tree may be available on different systems in different places; if the application needs to reference its distribution tree, this should be determined from the *application*_ROOT environment variable.

The application startup script arranges that the PATH and *application*_ROOT environment variables are set correctly while the application is running. If the application's distribution tree (placed into **/vol/***application* or **/usr/local/***application*) does not have an executable binary with the name of the application (for example, **/vol/***application***/bin.***arch***/***application*), then **start_applic** can not be used, and a customized application startup script must be used instead. Such scripts must also allow users to invoke the application from systems of any architecture, without requiring them to customize their own environments.

Note that there are two contrasting models of software installation. The **heterogeneous model** assumes general availability of the software, and solves the "which binaries to use" problem with no administrative overhead. The **homogeneous model** assumes very limited availability of software, requires administrative procedures to ensure that **/usr/local** only contains binaries of the local architecture, and does not really account for networked installations. It is easier to add support for additional architectures using a heterogeneous network model of software installation from the beginning.

Heterogeneous Networked Installations

Applications available on the network are available through **/vol/***application* and exported either to all systems or just to selected ones, as licensing restrictions allow. The export point is **/export/vol/***application*, which is a symbolic link to the actual installation point, typically the **/files/vol/***application* directory. All subdirectories not explicitly tagged with a processor architecture are shared among all processor architectures; thus while the **.../bin.sun386** and **.../lib.sun386** subdirectories contain, respectively, binaries and libraries executable only on systems of the Sun386i architecture, the **.../bin** directory contains executables that run on any architecture (typically using an interpreter such as **/bin/sh**), and the **.../etc** directory only contains sharable configuration files.

Homogeneous Single Machine Installations

Applications available only on a specific machine and its boot clients of the same architecture are installed into **/usr/local/***application*. This directory supports only a single architecture, so **/usr/local/***application***/bin** contains binaries executable only on the local architecture, and **/usr/local/***application***/lib** contains libraries executable only on the local architecture. Any sharable files are grouped in **/usr/local/***application***/share**.

To install an application onto a boot server to serve boot clients with other architectures, place the application in **/usr/local/***application* on the clients, as described above. The installation point (on the server) for application binaries of architecture *arch* is **/export/local/***arch***/***application*. When the architecture is the server architecture, this case is identical with the one above.

Other Installations

Smaller applications (of only one or two files) may be installed into the appropriate **/vol/local/bin.***arch* directory, or possibly into **/export/local/***arch***/bin**. These directories are in user's default paths, so the application does not need to be registered using **start_applic**.

FILES

/files<n>/vol/*application*
/export/vol/*application*
/vol/*application*
/vol/*application***/bin.***arch***/application**
/usr/local/*application*
/export/local/*arch/application*

SEE ALSO

auto.vol(5), **exports**(5), **automount**(8), **exportfs**(8)

Sun386i SNAP Administration

Sun386i Advanced Administration

NAME

statd, rpc.statd – network status monitor

SYNOPSIS

/usr/etc/rpc.statd

DESCRIPTION

statd is an intermediate version of the status monitor. It interacts with **lockd**(8C) to provide the crash and recovery functions for the locking services on NFS.

FILES

/etc/sm
/etc/sm.bak
/etc/state

SEE ALSO

statmon(5), **lockd**(8C)

BUGS

The crash of a site is only detected upon its recovery.

NAME

sticky – mark files for special treatment

DESCRIPTION

The *sticky bit* (file mode bit 01000, see **chmod**(2V)) is used to indicate special treatment of certain files and directories. A directory for which the sticky bit is set restricts deletion of files it contains. A file in a sticky directory may only be removed or renamed by a user who has write permission on the directory, and either owns the file, owns the directory, or is the super-user. This is useful for directories such as **/tmp**, which must be publicly writable, but should deny users permission to arbitrarily delete or rename the files of others.

If the sticky bit is set on a regular file and no execute bits are set, the system's page cache will not be used to hold the file's data. This bit is normally set on swap files of diskless clients so that accesses to these files do not flush more valuable data from the system's cache. Moreover, by default such files are treated as swap files, whose inode modification times may not necessarily be correctly recorded on permanent storage.

Any user may create a sticky directory. See **chmod** for details about modifying file modes.

BUGS

mkdir(2V) will not create a file with the sticky bit set.

FILES

/tmp

SEE ALSO

chmod(1V), **chmod**(2V), **chown**(2V), **mkdir**(2V)

NAME

sundiag – system diagnostics

SYNOPSIS

/usr/diag/sundiag/sundiag [**–Cmt**] [**–k** *kernel_name*] [**–o** *saved_options_file*]
[*generic_tool_arguments*]

AVAILABILITY

This program is available with the *User Diagnostics* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

sundiag is a diagnostic facility that tests the functionality of the operating system and reports its findings. It can also be used to report the hardware configuration as detected by the system.

You must be **root** to use **sundiag**.

When run on the console monitor, **sundiag** takes full advantage of the *SunView 1* windowing environment. There are four subwindows:

- A control panel for displaying the discovered hardware configuration and manipulating of the numerous test parameters and options.
- A test status panel which shows the test results.
- A console window which is used to display messages.
- A performance monitor.

There are also some popup frames, including a text frame for viewing **sundiag** and system log files.

When executed from a terminal, **sundiag** uses **curses**(3V) to simulate each subwindow on the screen.

sundiag consists of **sundiag**, along with several binary modules and executable files containing the actual test code, all of which reside in **/usr/diag/sundiag**.

OPTIONS

–C Redirect the console output from any existing console window to the **sundiag** console subwindow.

–m Create a device file for all devices found during the kernel probe. **sundiag** uses the same major/minor device numbers and permissions declared in **/dev/MAKEDEV**.

–t Run **sundiag** on a terminal.

–k *kernel_name*
Specify the customized kernel name that was used to boot up the system. The default kernel name is **/vmunix**. Since the **rstatd**(8C) that the performance monitor requires is hard-wired to use **/vmunix** as the kernel name, the performance monitor is disabled when this option is specified.

–o *saved_options_file*
Use the *saved_options_file* to restore options. The default option file is **.sundiag**. **.sundiag** is used if the **–o** option is not used and if the default file exists.

generic_tool_arguments
Refer to **sunview**(1) for examples of generic tool arguments that may be used with **sundiag**.

FILES

/var/adm/sundiaglog/options/.sundiag	start-up option file
/usr/diag/sundiag/.usertest	user-defined test description file

SEE ALSO

sunview(1), **curses**(3V), **rstatd**(8C)

Installing SunOS 4.1
Sundiag User's Guide

NAME

suninstall – install and upgrade the SunOS operating system

SYNOPSIS

/usr/etc/install/suninstall

DESCRIPTION

suninstall is a forms-based subsystem for installing and upgrading the SunOS operating system. Unlike previous installation subsystems, **suninstall** does not require recapitulation of an interrupted procedure; you can pick up where you left off. A new invocation of **suninstall** displays the saved information and offers the user an opportunity to make any needed alterations before it proceeds.

Note: **suninstall** only exists in the mini-root and should only be invoked from there (see *Installing SunOS 4.1*).

suninstall allows installation of the operating system onto any system configuration, be it standalone, dataless, a homogeneous file server, or a heterogeneous server. It installs the various versions of the operating system needed by clients on a heterogeneous file server, from any Sun distribution media format. The number of different system versions that can be installed is only limited to the disk space available.

After the initial installation, the suninstall utility program **add_client**(8) adds clients while the server is running in multiuser mode. The suninstall **add_services**(8) program converts a standalone system or server into a heterogeneous file server, without rebooting, while the system is running in multiuser mode. To remove a diskless client, use the suninstall **rm_client**(8) program in multiuser mode.

To abort the installation procedure, use the interrupt character (typically CTRL-C).

USAGE

Refer to *Installing SunOS 4.1* for more information on the various menus and selections.

FILES

/usr/etc/install	directory containing installation programs and scripts
/usr/etc/install/xdrtoc	subsystem utility program
/etc/install	directory containing suninstall data files

SEE ALSO

add_client(8), **add_services**(8), **extract_unbundled**(8), **rm_client**(8)

Installing SunOS 4.1

NOTES

It is advisable to exit **suninstall** through the exit options from the **suninstall** menus.

NAME

swapon – specify additional device for paging and swapping

SYNOPSIS

/usr/etc/swapon –a

/usr/etc/swapon *name...*

DESCRIPTION

swapon specifies additional devices or files on which paging and swapping are to take place. The system begins by swapping and paging on only a single device so that only one disk is required at bootstrap time. Calls to **swapon** normally occur in the system multi-user initialization file **/etc/rc** making all swap devices available, so that the paging and swapping activity is interleaved across several devices.

The second form gives individual block devices or files as given in the system swap configuration table. The call makes only this space available to the system for swap allocation.

Note: "swap files" made with **mkfile**(8) can be used as swap areas over NFS.

OPTIONS

–a Make available all devices of type **swap** in **/etc/fstab**. Using **swapon** with the **–a** option is the normal usage.

FILES

/dev/sd?b	
/dev/xy?b	
/dev/xd?b	normal paging devices
/etc/fstab	
/etc/rc	

SEE ALSO

swapon(2), **fstab**(5), **init**(8), **mkfile**(8)

BUGS

There is no way to stop paging and swapping on a device. It is therefore not possible to make use of devices which may be dismounted during system operation.

NAME

sys-config – configure a system or administer configuration information

SYNOPSIS

/usr/etc/install/sys-config

DESCRIPTION

sys-config "unpacks" a machine and sets up its configuration. **sys-config** automatically runs when a pre-installed system is booted for the first time. It should not be run by hand. Instead, run **sys-unconfig**(8) to return the system to its pre-installed state. Then, reboot system, which will run **sys-config** automatically.

A system's configuration consists of hostname, Network Interface Service (NIS) domain name, timezone and IP address.

sys-config does the following:

- Edits the **/etc/hosts** with the correct hostname and IP address.
- Sets the hostname in **/etc/rc.boot**.
- Sets the domainname in **/etc/rc.single**.
- Sets the **/usr/lib/zoneinfo/localtime** file.
- Enables the Network Information Service (NIS) if the NIS service was requested.

When **sys-config** is finished, it prompts for a system reboot.

The default answer to any particular question is the current value of that configuration parameter. Parameters that have not changed can be quickly skipped over to get to the one that should be changed by typing a RETURN.

sys-config is potentially a dangerous utility and can be run only by the super-user.

FILES

/etc/hosts
/usr/lib/zoneinfo/localtime
/usr/etc/install/sys_info

SEE ALSO

sys-unconfig(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

sys-unconfig – undo a system's configuration

SYNOPSIS

/usr/etc/install/sys-unconfig

DESCRIPTION

sys-unconfig packs up a machine to make it ready to be configured again.

It restores a systems's configuration to an "as-manufactured" state. A system's configuration consists of hostname, Network Interface Service (NIS) domain name, timezone and IP address.

sys-unconfig does the following:

- Restores the default **/etc/hosts** file.
- Removes the default hostname in **/etc/hostname.??[0-9]**.
- Removes the default domainname in **/etc/defaultdomain**.
- Removes the default **/usr/lib/zoneinfo/localtime** file.
- Disables the Network Information Service (NIS) if the NIS service was requested.

When **sys-unconfig** is finished, it will prompt for a system shutdown.

sys-unconfig is potentially a dangerous utility and can only be run by the super-user.

FILES

/etc/hosts
/usr/lib/zoneinfo/localtime
/usr/etc/install/sys_info

SEE ALSO

sys-config(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

syslogd – log system messages

SYNOPSIS

/usr/etc/syslogd [**–d**] [**–f***configfile*] [**–m** *interval*]

DESCRIPTION

syslogd reads and forwards system messages to the appropriate log files and/or users, depending upon the priority of a message and the system facility from which it originates. The configuration file **/etc/syslog.conf** (see **syslog.conf**(5)) controls where messages are forwarded. **syslogd** logs a mark (timestamp) message every *interval* minutes (default 20) at priority **LOG_INFO** to the facility whose name is given as **mark** in the **syslog.conf** file.

A system message consists of a single line of text, which may be prefixed with a priority code number enclosed in angle-brackets (<>); priorities are defined in **sys/syslog.h.**

syslogd reads from the **AF_UNIX** address family socket **/dev/log**, from an Internet address family socket specified in **/etc/services**, and from the special device **/dev/klog** (for kernel messages).

syslogd reads the configuration file when it starts up, and again whenever it receives a HUP signal, at which time it also closes all files it has open, re-reads its configuration file, and then opens only the log files that are listed in that file. **syslogd** exits when it receives a **TERM** signal.

As it starts up, **syslogd** creates the file **/etc/syslog.pid**, if possible, containing its process ID (PID).

Sun386i DESCRIPTION

syslogd translates messages using the databases specified on an optional line in the **syslog.conf** as indicated with a **translate** entry.

The format of these databases is described in **translate**(5).

OPTIONS

–d	Turn on debugging.
–f*configfile*	Specify an alternate configuration file.
–m *interval*	Specify an interval, in minutes, between mark messages.

FILES

/etc/syslog.conf	configuration file
/etc/syslog.pid	process ID
/dev/log	AF_UNIX address family datagram log socket
/dev/klog	kernel log device
/etc/services	network services database

SEE ALSO

logger(1), **syslog**(3), **syslog.conf**(5), **translate**(5)

NAME

talkd, in.talkd – server for talk program

SYNOPSIS

/usr/etc/in.talkd

DESCRIPTION

talkd is a server used by the **talk**(1) program. It listens at the udp port indicated in the "talk" service description; see **services**(5). The actual conversation takes place on a tcp connection that is established by negotiation between the two machines involved.

SEE ALSO

talk(1), **services**(5), **inetd**(8C)

BUGS

The protocol is architecture dependent, and can not be relied upon to work between Sun systems and other machines.

NAME

telnetd, in.telnetd – TCP/IP TELNET protocol server

SYNOPSIS

/usr/etc/in.telnetd

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

telnetd is a server which supports the TCP/IP standard TELNET virtual terminal protocol. **telnetd** is invoked by the internet server (see **inetd**(8C)), normally for requests to connect to the TELNET port as indicated by the **/etc/services** file (see **services**(5)).

telnetd operates by allocating a pseudo-terminal device (see **pty**(4)) for a client, then creating a login process which has the slave side of the pseudo-terminal as its standard input, output, and error. **telnetd** manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session is started up, **telnetd** sends TELNET options to the client side indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal type information* from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with **XTABS**, **ICRNL**, and **ONLCR** enabled (see **termio**(4)).

telnetd is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*. **telnetd** is willing to have the remote client *do*: *binary*, *terminal type*, and *suppress go ahead*.

SEE ALSO

telnet(1C)

Postel, Jon, and Joyce Reynolds, "Telnet Protocol Specification," RFC 854, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

BUGS

Some TELNET commands are only partially implemented.

The TELNET protocol allows for the exchange of the number of lines and columns on the user's terminal, but **telnetd** doesn't make use of them.

Because of bugs in the original 4.2 BSD **telnet**(1C), **telnetd** performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD **telnet**(1C).

Binary mode has no common interpretation except between similar operating systems

The terminal type name received from the remote client is converted to lower case.

The *packet* interface to the pseudo-terminal (see **pty**(4)) should be used for more intelligent flushing of input and output queues.

telnetd never sends TELNET *go ahead* commands.

telnetd can only support 64 pseudo-terminals.

NAME

tfsd – TFS daemon

SYNOPSIS

/usr/etc/tfsd

DESCRIPTION

tfsd is the daemon for the Translucent File Service (TFS). This daemon is started by **inetd**(8C) whenever a TFS request is made.

tfsd looks up a file by looking in the frontmost directory (see **tfs**(4S)). If the file is not found in this directory, **tfsd** follows the *searchlink* from the frontmost directory to the directory immediately behind it. **tfsd** continues to search for the file until one of the following conditions is met:

- The file is found in a directory.
- There are no more searchlinks to follow.
- A *whiteout* entry for the file is found.

The searchlinks and whiteout entries are specified in **.tfs_info** files.

FILES

.tfs_info	holds searchlink and whiteout entries

SEE ALSO

unwhiteout(1), **lsw**(1), **tfs**(4S), **mount_tfs**(8)

NAME

tftpd, in.tftpd – TCP/IP Trivial File Transfer Protocol server

SYNOPSIS

/usr/etc/in.tftpd [–s] [*homedir*]

Sun386i SYNOPSIS

/usr/etc/in.tftpd [–s] [–p] [*homedir*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

tftpd is a server that supports the TCP/IP Trivial File Transfer Protocol (TFTP). This server is normally started by **inetd**(8C) and operates at the port indicated in the **tftp** Internet service description in the **/etc/inetd.conf** file; see **inetd.conf**(5) for details.

Before responding to a request, the server attempts to change its current directory to *homedir*; the default value is **/tftpboot**.

Sun386i DESCRIPTION

The **tftpd** daemon acts as described above, except that it will perform certain filename mapping operations unless instructed otherwise by the **–p** command line argument or when operating in a secure environment. This mapping affects only TFTP boot requests and will not affect requests for existing files.

The semantics of the changes are as follows. Only filenames of the format *ip-address* or *ip-address.arch*, where *ip-address* is the IP address in hex, and *arch* is the hosts's architecture (as returned by the **arch**(1) command), that do not correspond to files in **/tftpboot**, are mapped. If the address is known through a Network Interface Service (NIS) lookup, any file of the form **/tftpboot/ip-address*** (with or without a suffix) is returned. If there are multiple such files, any one may be returned. If the *ip-address* is unknown (that is if the **ipalloc (8C)** service says the name service does not know the address), the filename is mapped as follows: Names without the *arch* suffix are mapped into the name **pnp.SUN3**, and names with the suffix are mapped into **pnp.** *arch*. That file is returned if it exists.

OPTIONS

–s Secure. When specified, the directory change must succeed; and the daemon also changes its root directory to *homedir*.

The use of **tftp** does not require an account or password on the remote system. Due to the lack of authentication information, **tftpd** will allow only publicly readable files to be accessed. Files may be written only if they already exist and are publicly writable. Note: this extends the concept of "public" to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling this service.

tftpd runs with the user ID (UID) and group ID (GID) set to **–2**, under the assumption that no files exist with that owner or group. However, nothing checks this assumption or enforces this restriction.

Sun386i OPTIONS

–p Disable pnp entirely. Do not map filenames.

Sun386i FILES

/tftpboot/* filenames are IP addresses

SEE ALSO

tftp(1C) **inetd**(8C), **ipallocd**(8C), **netconfig**(8C)

Sollins, K.R., *The TFTP Protocol (Revision 2)*, RFC 783, Network Information Center, SRI International, Menlo Park, Calif., June 1981.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

Sun386i WARNINGS

A request for an *ip-address* from a Sun-4 can be satisfied by a file named *ip-address*.**386** for compatibility with some early Sun-4 PROM monitors.

NAME

tic – terminfo compiler

SYNOPSIS

tic [–**v**[*n*]] [–**c**] *filename*

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

tic compiles a **terminfo**(5V) source file into the compiled format. The results are placed in the directory **/usr/share/lib/terminfo**. The compiled format is used by the **curses**(3V) library.

Each entry in the file describes the capabilities of a particular terminal. When a **use**=*entry* field is given in a terminal entry, **tic** reads in the binary (compiled) description of the indicated *entry* from **/usr/share/lib/terminfo** to duplicate the contents of that entry within the one being compiled. However, if an *entry* by that name is specified in *filename*, the entry in that source file is used first. Also, if a capability is defined in both entries, the definition in the current entry's source file is used.

If the environment variable **TERMINFO** is set, that directory is searched and written to instead of **/usr/share/lib/terminfo**.

OPTIONS

–**v**[*n*]
: Verbose. Display trace information on the standard error. The optional integer argument is a number from 1 to 10, inclusive, indicating the desired level of detail. If *n* is omitted, the default is 1.

–**c**
: Only check *filename* for errors. Errors in **use**= links are not detected.

FILES

/usr/share/lib/terminfo/?/* compiled terminal description data base

SEE ALSO

fork(2V), **curses**(3V), **curses**(3V), **malloc**(3V), **term**(5), **terminfo**(5V)

BUGS

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 1024 bytes.

When the –**c** option is used, duplicate terminal names will not be diagnosed; however, when –**c** is not used, they will be.

For backward compatibility, cancelled capabilities will not be marked as such within the terminfo binary unless the entry name has a '+' within it. Such terminal names are only used for inclusion with a **use**= field, and typically aren't used for actual terminal names.

DIAGNOSTICS

Most diagnostic messages produced by **tic** are preceded with the approximate line number and the name of the entry being processed.

mkdir *name* **returned bad status**
: The named directory could not be created.

File does not start with terminal names in column one
: The first thing seen in the file, after comments, must be the list of terminal names.

Token after a seek(2) not NAMES
: Somehow the file being compiled changed during the compilation.

Not enough memory for use_list element
Out of memory
Not enough free memory was available (**malloc**(3V) failed).

Can't open *filename*
The named file could not be opened or created.

Error in writing *filename*
The named file could not be written to.

Can't*link* **filename** *to* **filename**
A link failed.

Error in re-reading compiled *filename*
The compiled file could not be read back in.

Premature EOF
The current entry ended prematurely.

Backspaced off beginning of line
This error indicates something wrong happened within **tic**.

Unknown Capability – *filename*
The named invalid capability was found within the file.

Wrong type used for capability ...
For example, a string capability was given a numeric value.

Unknown token type
Tokens must be followed by '@' to cancel, ',' for booleans, '#' for numbers, or '=' for strings.
name: **bad term name**
Line *n*: **Illegal terminal name** – *name*
Terminal names must start with a letter or digit
The given name was invalid. Names must not contain white space or slashes, and must begin with a letter or digit.

name: **terminal name too long.**
An extremely long terminal name was found.

name: **terminal name too short.**
A one-letter name was found.

name **defined in more than one entry. Entry being used is** *name* .
An entry was found more than once.

Terminal name *name* **synonym for itself**
A name was listed twice in the list of synonyms.

At least one synonym should begin
At least one of the names of the terminal should begin with a letter.

Illegal character – **c**
The given invalid character was found in the input file.

Newline in middle of terminal name
The trailing comma was probably left off of the list of names.

Missing comma
A comma was missing.

Missing numeric value
The number was missing after a numeric capability.

NULL string value
The proper way to say that a string capability does not exist is to cancel it.

Very long string found. Missing comma?
Self-explanatory.

Unknown option. Usage is:
An invalid option was entered.

Too many file names. Usage is:
Self-explanatory.

name **non-existent or permission denied**
The given directory could not be written into.

name **is not a directory**
Self-explanatory.

name: **Permission denied**
Access denied.

name: **Not a directory**
tic wanted to use the given name as a directory, but it already exists as a file

SYSTEM ERROR!! Fork failed!!!
A **fork**(2V) failed.

Error in following up use-links.
Either there is a loop in the links or they reference non-existent terminals. The following is a list of the entries involved:
A **terminfo**(5V) entry with a **use**=*name* capability either referenced a non-existent terminal called *filename* or *filename* somehow referred back to the given entry.

NAME

tnamed, in.tnamed – TCP/IP Trivial name server

SYNOPSIS

/usr/etc/in.tnamed [–v]

DESCRIPTION

tnamed is a server that supports the TCP/IP Name Server Protocol. The name server operates at the port indicated in the "name" service description (see **services**(5)), and is invoked by **inetd**(8C) when a request is made to the name server.

Two known clients of this service are the MIT PC/IP software the Bridge boxes.

OPTIONS

–v Invoke the daemon in verbose mode.

SEE ALSO

uucp(1C), **services**(5), **inetd**(8C)

Postel, Jon, *Internet Name Server*, IEN 116, SRI International, Menlo Park, California, August 1979.

BUGS

The protocol implemented by this program is obsolete. Its use should be phased out in favor of the Internet Domain protocol. See **named**(8C).

NAME

trpt – transliterate protocol trace

SYNOPSIS

/usr/etc/trpt [**–afjst**] [**–p***hex-address*] [*system* [*core*]]

DESCRIPTION

trpt interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see **getsockopt**(2)), and prints a readable description of these records. When no options are supplied, **trpt** prints all the trace records found in the system grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

OPTIONS

–a In addition to the normal output, print the values of the source and destination addresses for each packet recorded.

–f Follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.

–j Just give a list of the protocol control block addresses for which there are trace records.

–s In addition to the normal output, print a detailed description of the packet sequencing information.

–t In addition to the normal output, print the values for all timers at each point in the trace.

–p *hex-address*
Show only trace records associated with the protocol control block, the address of which follows.

The recommended use of **trpt** is as follows. Isolate the problem and enable debugging on the **socket**(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the **–A** option to **netstat**(8C). Then run **trpt** with the **–p** option, supplying the associated protocol control block addresses. The **–f** option can be used to follow the trace log once the trace is located. If there are many sockets using the debugging option, the **–j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

FILES

/vmunix
/dev/kmem

SEE ALSO

getsockopt(2), **netstat**(8C)

DIAGNOSTICS

no namelist When the system image does not contain the proper symbols to find the trace buffer; others which should be self explanatory.

BUGS

Should also print the data for each input or output, but this is not saved in the trace record.

The output format is inscrutable and should be described here.

NAME

ttysoftcar – enable/disable carrier detect

SYNOPSIS

ttysoftcar [**–y** | **–n**] *tty* ...

ttysoftcar –a

DESCRIPTION

For each *tty* specified **ttysoftcar** changes the carrier detect flag using the **TIOCSSOFTCAR ioctl( )** request (see **tty**(4)). If the **–a** option is specified, **ttysoftcar** sets all **tty**'s in the **/etc/ttytab** file to the carrier detection mode specified by their status field. If this field is set to **local**, software carrier detection is turned on. If this field is set to anything other than **local**, as is usually the case for modems, software carrier detection is turned off. **ttysoftcar** ignores devices in the **/etc/ttytab** file which do not exist.

If no options are specified, **ttysoftcar** returns the current status for *tty*. This status is reported as **y** or **n**.

OPTIONS

–a Reset **tty**s to appropriate values based on the status field of the **/etc/ttytab** file.

–y Turn on software carrier detect.

–n Turn off software carrier detect. Use hardware carrier detect.

SEE ALSO

termio(4), **zs**(4S), **ttytab**(5)

NAME

tunefs – tune up an existing file system

SYNOPSIS

/usr/etc/tunefs [**–a** *maxcontig*] [**–d** *rotdelay*] [**–e** *maxbpg*] [**–m** *minfree*] *special* | *filesystem*

DESCRIPTION

tunefs is designed to change the dynamic parameters of a file system which affect the layout policies. The parameters which are to be changed are indicated by the OPTIONS given below:

OPTIONS

–a *maxcontig*

This specifies the maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see **–d** below). The default value is one, since most device drivers require an interrupt per disk transfer. Device drivers that can chain several buffers together in a single transfer should set this to the maximum chain length.

–d *rotdelay*

This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

–e *maxbpg*

This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to about one quarter of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

–m *minfree*

This value specifies the percentage of space held back from normal users; the minimum free space threshold. The default value used is 10%. This value can be set to zero, however up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. Note: if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

SEE ALSO

fs(5), **dumpfs**(8), **mkfs**(8), **newfs**(8)

System and Network Administration

BUGS

This program should work on mounted and active file systems. Because the super-block is not kept in the buffer cache, the program will only take effect if it is run on dismounted file systems; if run on the root file system, the system must be rebooted.

NAME

tzsetup – set up old-style time zone information in the kernel

SYNOPSIS

/usr/etc/tzsetup

DESCRIPTION

tzsetup attempts to find the offset from GMT and old-style Daylight Savings Time correction type (see **gettimeofday**(2)) that most closely matches the default time zone for the machine, and to pass this information to the kernel with a **settimeofday** ( ) call (see **gettimeofday**(2)). This is necessary if programs built under releases of SunOS prior to 4.0 are to be run; those programs get time zone information from the kernel using **gettimeofday**.

If it cannot find the offset from GMT, the offset is set to 0; if it cannot find the Daylight Savings Time correction type, it is set to **DST_NONE**, indicating that no Daylight Savings Time correction is to be performed.

DIAGNOSTICS

tzsetup: Can't open /usr/share/lib/zoneinfo/localtime: *reason*
The time zone file for the current time zone could not be opened.

tzsetup: Error reading /usr/lib/zoneinfo/localtime: *reason*
The time zone file for the current time zone could not be read.

tzsetup: Two or more time zone types are equally valid — no DST selected
There were two or more Daylight Savings Time correction types that generated results that were equally close to the correct results. None of them was selected. Programs built under versions of SunOS prior to 4.0 may not convert dates correctly.

tzsetup: No old-style time zone type is valid — no DST selected
None of the Daylight Savings Time correction types generated results that were in any way correct; none of them was selected. Programs built under versions of SunOS prior to 4.0 may not convert dates correctly.

tzsetup: Warning: No old-style time zone type is completely valid
None of the Daylight Savings Time correction types generated results that were completely correct; the best of them was selected. Programs built under versions of SunOS prior to 4.0 may not convert dates correctly.

tzsetup: Can't set time zone
tzsetup was run by a user other than the super-user; only the super-user may change the kernel's notion of the current time zone.

SEE ALSO

gettimeofday(2), **tzfile**(5), **zic**(8)

NAME

uid_allocd, gid_allocd – UID and GID allocator daemons

SYNOPSIS

/usr/etc/rpc.uid_allocd
/usr/etc/rpc.gid_allocd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The UID (or GID) allocator will temporarily allocate an unused UID (or GID) for use by account administration tools. It maintains a cache of UIDs (GIDs) that have been allocated by potentially multiple tools (or instances of tools) in a distributed system, so that they can create accounts (or groups) concurrently. It also provides the ability to safely enter a UID (GID) into the cache which was allocated using some other method, such as manually by an administrator; and the ability to delete entries from the cache. Entries in this cache persist for at least an hour even through system crashes.

These allocators are available on the system which contains the master copy of the list of UIDs (or GID). Since this list is currently maintained using the Network Interface Service (NIS), the service is available on the master of the **passwd.byuid** (**group.bygid**) NIS map. The service could be provided using a UID database service other than the NIS service.

This implementation uses DES authentication (the Sun Secure RPC protocol) to restrict access to this function. The only clients privileged to allocate UIDs (GIDs) are those whose net IDs are in the *accounts* group (fixed at GID 11). All machine IDs are allowed to allocate UIDs (GIDs).

If the file **/etc/ugid_alloc.range** exists, the allocator only allocates UIDs (GIDs) in the range listed there. This feature is intended to be used by sites which have multiple NIS domains on their networks; each NIS domain would be assigned a unique range of UIDs (GIDs). If the file exists, and the local NIS domain is not explicitly assigned a unique range of UIDs or GID, none will be allocated. Without a mechanism to ensure that UIDs are uniquely assigned between NIS domains that share resources, normal NFS security mechanisms (excluding Secure NFS) may fail to serve as an advisory security mechanism. Common alternative methods for ensuring UID uniqueness include using a function of some preexisting identifier such as an employee number, or using a single NIS domain for the entire site.

FILES

/var/yp/*domainname***/passwd.byuid.{dir,pag}**
/var/yp/*domainname***/group.bygid.{dir,pag}**
/var/yp/*domainname***/netid.byname.{dir,pag}**
/etc/uid_alloc.cache
/etc/gid_alloc.cache
/etc/ugid_alloc.range
/usr/include/rpcsvc/uid_alloc.x
/usr/include/rpcsvc/gid_alloc.x

SEE ALSO

snap(1), **ugid_alloc.range**(5), **logintool**(8)

BUGS

Using UID (GID) ranges does not solve the problem that two different machines, or groups of machines, may assign different meaning to a given UID (GID).

The current implementation of the daemon is tuned towards small lists of active UIDs (GIDs), both in the NIS service and in the cache it maintains.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

unadv – unadvertise a Remote File Sharing resource

SYNOPSIS

unadv *resource*

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

unadv unadvertises a Remote File Sharing (RFS) *resource*, which is the advertised symbolic name of a local directory, by removing it from the advertised information on the domain name server. **unadv** prevents subsequent remote mounts of that resource. It does not affect continued access through existing remote or local mounts.

An administrator at a server can unadvertise only those resources that physically reside on the local machine. A domain administrator can unadvertise any resource in the domain from the primary name server by specifying *resource* name as *domain.resource*. A domain administrator should only unadvertise another hosts resources to clean up the domain advertise table when that host goes down. Unadvertising another host's resource changes the domain advertise table, but not the host advertise table.

This command is restricted to the super-user.

If *resource* is not found in the advertised information, an error message will be sent to standard error.

SEE ALSO

adv(8), **fumount**(8), **nsquery**(8)

NAME

unconfigure – reset the network configuration for a Sun386i system

SYNOPSIS

/usr/etc/unconfigure [**–y**]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

unconfigure restores most of the system configuration and status files to the state they were in when delivered by Sun Microsystems, Inc. It also deletes all user accounts (including home directories), Network Interface Service (NIS) information, and any diskless client configurations that were set up.

After running **unconfigure,** a system halts. Rebooting it to multi-user mode at this point will start automatic system installation.

unconfigure is intended for use in the following situations:

- As one of the final steps in Software Manufacturing.
- In systems being set up with temporary configurations, holding no user accounts or diskless clients. These will occur during demonstrations and evaluation trials.
- To allow systems that had been used as standalones to be upgraded to join a network in a role other than as a master server. (See instructions later.)

unconfigure is potentially a dangerous utility; it does not work unless invoked by the super-user. As a warning, unless the **–y** option is passed, it will require confirmation that all user files and system software configuration information is to be deleted.

This utility is *not* recommended for routine use of any sort.

Resetting Temporary Configurations

If users need to set up and tear down configurations, **unconfigure** can be used to restore the system to an essentially as-manufactured state. The main concern here is that user accounts will be deleted, so this should not be done casually.

To reset a temporary configuration, just become the super-user and invoke **unconfigure.**

Upgrading Standalones to Network Clients

Systems that are going to be networked should be networked from the very first, if at all possible. This eliminates whole classes of compatibility problems, such as pathnames and (in particular) user account clashes.

Automatic system installation directly supports upgrading a single standalone system to an NIS master, and joining any number of unused systems (or systems upon which **unconfigure** has been run) into a network.

However, in the situation where standalone systems that have been used extensively are to be joined to a network, **unconfigure** can be used in conjunction with automatic system installation by a knowledgeable super-user to change a system's configuration from standalone to network client. This procedure is not recommended for use by inexperienced administrators.

The following procedure is not needed unless user accounts or other data need to be preserved; it is intended to ensure that every UID and GID is changed so as not to clash with those in use on the network. It must be applied to each system that is being upgraded from a standalone to a network client.

The procedure is as follows:

- Identify all accounts and files that you will want to save. If there are none, just run **unconfigure** and install the system on the network. Do not follow the remaining steps.
- Copy **/etc/passwd** to **/etc/passwd.bak.**

- Rename all the files (including home directories) so that they aren't deleted. (See **FILES** below.) These will probably be only in **/export/home.**
- Run **unconfigure** and install the system on the network.
- For each account listed in **/etc/passwd.bak** that you want to save, follow this procedure:
  - Create a new account on the network; if the UID and GID are the same as in **/etc/passwd.bak** on the standalone, then skip the next step. However, be sure that you do not make two different accounts with the same UID.
  - Use the **'chown –R'** command to change the ownership of the home directories.
  - You may need to rename the files you just chowned above, for example to ensure that they are the user's home directory. This may involve updating the **auto.home**(5) and **auto.home**(5) NIS maps, as well.
- Delete **/etc/passwd.bak.**

FILES

unconfigure deletes the following files, if they are present, replacing some of them with the distribution version if one is supposed to exist:

/etc/.rootkey	**/etc/ethers**	**/etc/localtime**	**/etc/publickey**
/etc/auto.home	**/etc/exports**	**/etc/net.conf**	**/etc/sendmail.cf**
/etc/auto.vol	**/etc/fstab**	**/etc/netmasks**	**/etc/syslog.conf**
/etc/bootparams	**/etc/group**	**/etc/networks**	**/etc/systems**
/etc/bootservers	**/etc/hosts**	**/etc/passwd**	**/single/ifconfig**
/var/sysex/*			

and all files in **/var/yp** except those distributed with the operating system.

unconfigure truncates all files in **/var/adm.** All user home directories in **/export/home** are deleted, except those for the default user account **users**, which is shipped with the operating system. All diskless client configuration information stored in **/export/roots**, **/export/swaps**, and **/export/dumps** is deleted.

SEE ALSO

chgrp(1), **find**(1), **group**(5), **passwd**(5) **adduser**(8), **chown**(8)

BUGS

More of the system configuration files should be reset.

This does not yet support taking a workstation off the network temporarily, for example, to take it home over the weekend for use as a standalone, or to move it to another network while traveling. This should be the default behavior.

The procedure for upgrading standalones to network clients should be automated; currently, only upgrading a standalone to a master server is automated.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

update – periodically update the super block

SYNOPSIS

/usr/etc/update

DESCRIPTION

update is a program that executes the **sync**(2) primitive every 30 seconds. This insures that the file system is fairly up to date in case of a crash. This command should not be executed directly, but should be executed out of the initialization shell command file.

SEE ALSO

sync(1), **sync**(2), **init**(8)

NAME

user_agentd – user agent daemon

SYNOPSIS

/usr/etc/rpc.user_agentd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

rpc.user_agentd is the remote service used by **snap**(1) to create, move, or delete home directories, and by the New User Accounts feature of **logintool**(8) to create new home directories. The user_agent daemon is normally invoked by **inetd**(8C), and runs on all non-diskless systems.

When creating a new home directory, the user_agent daemon executes the **copy_home**(8) script which resides in the home directory of the primary group to which a new user will be added.

SEE ALSO

snap(1), **copy_home**(8), **inetd**(8C), **logintool**(8)

NAME

uucheck – check the UUCP directories and Permissions file

SYNOPSIS

/usr/lib/uucp/uucheck [**–v**] [**–x** *debug_level*]

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uucheck checks for the presence of the UUCP system required files and directories. It also checks for some obvious errors in the **Permissions** file (**/usr/lib/uucp/Permissions**).

Note: **uucheck** can only be used by the super-user or **uucp**.

OPTIONS

–v Give a detailed explanation of how the UUCP programs will interpret the **Permissions** file.

–x *debug_level*

Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/etc/uucp/Maxuuscheds
/etc/uucp/Maxuuxqts
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **uucico**(8C), **uusched**(8C)

BUGS

The program does not check file/directory modes or some errors in the **Permissions** file such as duplicate login or machine name.

NAME

uucico – file transport program for the UUCP system

SYNOPSIS

/usr/lib/uucp/uucico [**–r** *role_number*] [**–x** *debug_level*] [**–i** *interface*] [**–d** *spool_directory*]
–s *system_name*

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uucico is the file transport program for UUCP work file transfers. **uux**(1C) and **uucp**(1C) both queue jobs that will be transferred by **uucico**. It is normally started by the scheduler, **uusched (8C),** but can be started manually; this is done for debugging. For example, the script **Uutry** starts **uucico** with debugging turned on.

OPTIONS

–r *role_number*

Specify the role that **uucico** should perform. *role_number* is the digit 1 for master mode or 0 for slave mode (default). Master mode should be specified when **uucico** is started by a program or **cron**(8).

–x *debug_level*

Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

–i *interface*

Define the interface used with **uucico**. This interface only affects slave mode. Known interfaces are UNIX (default).

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/etc/uucp/Devconfig
/etc/uucp/Sysfiles
/etc/uucp/Maxuuxqts
/etc/uucp/Maxuuscheds
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **cron**(8), **uusched**(8C)

NAME

uuclean – uucp spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uuclean [**–m**] [**–d***directory*] [**–n***time*] [**–p***pre*]

DESCRIPTION

uuclean scans the spool directory for files with the specified prefix and deletes all those which are older than the specified number of hours.

OPTIONS

–d*directory*
Clean the indicated spool directory.

–m Send mail to the owner of the file when it is deleted.

–n*time* Files whose age is more than *time* hours are deleted if the prefix test is satisfied (default time is 72 hours).

–p*pre* Scan for files with *pre* as the file prefix. Up to 10 **–p** arguments may be specified. A **–p** without any *pre* following deletes all files older than the specified time.

uuclean will typically be started by **cron**(8).

FILES

/usr/lib/uucp	directory with commands used by uuclean internally
/usr/lib/uucp/spool	spool directory

SEE ALSO

uucp(1C), **uux**(1C), **cron**(8)

NAME

uucleanup – UUCP spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uucleanup [**–C***time*] [**–D***time*] [**–m***string*] [**–o***time*] [**–s***system*] [**–W***time*] [**–x** *debug_level*] [**–X***time*]

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uucleanup will scan the spool directories for old files and take appropriate action to remove them in a useful way:

- Inform the requestor of send/receive requests for systems that cannot be reached.
- Return mail, which cannot be delivered, to the sender.
- Delete or execute **rnews** for **rnews** type files (depending on where the news originated — locally or remotely).
- Remove all other files.

In addition, there is provision to warn users of requests that have been waiting for a given number of days (default 1 day). Note: **uucleanup** will process as if all option *time*s were specified to the default values unless *time* is specifically set.

This program is typically started by the shell **uudemon.cleanup**, which should be started by **cron**(8).

OPTIONS

–C*time* Remove any **C.** files that are at least *time* days old (default 7 days), and send appropriate information to the requestor.

–D*time* Remove any **D.** files that are at least *time* days old (default 7 days), and make an attempt to deliver mail messages and execute **rnews** when appropriate.

–m*string*
Include this line in the warning message generated by the **–W** option. The default line is '**See your local administrator to locate the problem**'.

–o*time* Delete other files that are more than *time* days old (default 2 days).

–s*system*
Execute for the spool directory for the remote system *system* only.

–W*time*
Send a mail message to be sent to the requestor warning about the delay in contacting the remote for any **C.** files that are *time* days old (default 1 day). The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling whom to call to check the problem (**–m** option).

–x *debug_level*
Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

–X*time* Remove any **X.** files that are at least *time* days old (default 2 days). The **D.** files are probably not present (if they were, the **X.** could get executed). But if there are **D.** files, they will be taken care of by **D.** processing.

FILES

/usr/lib/uucp	directory with commands used by **uucleanup** internally
/var/spool/uucp	spool directory

SEE ALSO

uucp(1C), **uux**(1C), **cron**(8)

NAME

uucpd – UUCP server

SYNOPSIS

/usr/etc/in.uucpd

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uucpd is the server for supporting UUCP connections over networks.

uucpd is invoked by **inetd**(8C) when a UUCP connection is established (that is, a connection to the port indicated in the "uucp" service specification; see **services**(5)), and executes the following protocol:

1) The server prompts with **login:**. The **uucico**(8C) process at the other end must supply a username.

2) Unless the username refers to an account without a password, the server then prompts with **Password:**. The **uucico** process at the other end must supply the password for that account.

If the username is not valid or is valid but refers to an account that does not have **/usr/lib/uucp/uucico** as its login shell, or if the password is not the correct password for that account, the connection is dropped. Otherwise, **uucico** is run, with the user ID, group ID, group set, and home directory for that account, with the environment variables **USER** and **LOGNAME** set to the specified username, and with a **–u** flag specifying the username. Entries are made in **/var/adm/wtmp** and **/var/adm/lastlog** for the username.

FILES

/var/adm/wtmp	accounting
/var/adm/lastlog	time of last login

SEE ALSO

services(5), **inetd**(8C), **uucico**(8C)

DIAGNOSTICS

All diagnostic messages are returned on the connection, after which the connection is closed.

user read

An error occurred while reading the username.

passwd read

An error occurred while reading the password.

Login incorect.

The username is invalid or refers to an account with a login shell other than **/usr/lib/uucp/uucico**, or the password is not the correct password for the account.

NAME

uusched – the scheduler for the UUCP file transport program

SYNOPSIS

/usr/lib/uucp/uusched [**–u** *debug_level*] [**–x** *debug_level*]

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uusched is the UUCP file transport scheduler. It is usually started by the daemon **uudemon.hour** that is started by **cron**(8) from an entry in the system **crontab** file:

39 * * * * /bin/su uucp –c "/usr/lib/uucp/uudemon.hour > /dev/null"

OPTIONS

–u *debug_level*

Pass *debug_level* as '**–x** *debug_level*' to any invocations of **uucico**(8C) started by **uusched**.

–x *debug_level*

Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **cron**(8), **uucico**(8C)

NAME

uuxqt – execute remote command requests

SYNOPSIS

/usr/lib/uucp/uuxqt [**–x** *debug_level*]

DESCRIPTION

uuxqt is the program that executes remote job requests from remote systems generated by the use of the **uux**(1C) command. **mail**(1) uses **uux** for remote mail requests. **uuxqt** searches the spool directories looking for **X.** files. For each **X.** file, **uuxqt** checks to see if all the required data files are available and accessible, and file commands are permitted for the requesting system. The **Permissions** file is used to validate file accessibility and command execution permission.

OPTIONS

–x *debug_level*

Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

ENVIRONMENT

There are two environment variables that are set before the **uuxqt** command is executed:

UU_MACHINE Machine that sent the job (the previous one).

UU_USER User that sent the job.

These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

FILES

/etc/uucp/Permissions
/etc/uucp/Maxuuxqts
/var/spool/uucp/*
/var/spool/locks/LCK*

SEE ALSO

mail(1), **uucp**(1C), **uustat**(1C), **uux**(1C), **uucico**(8C)

NAME

vipw – edit the password file

SYNOPSIS

/usr/etc/vipw

DESCRIPTION

vipw edits the password file while setting the appropriate locks, and does any necessary processing after the password file is unlocked. If the password file is already being edited, then you will be told to try again later. The **vi**(1) editor will be used unless the environment variable **VISUAL** or **EDITOR** indicates an alternate editor.

vipw performs a number of consistency checks on the password entry for root, and will not allow a password file with a "mangled" root entry to be installed. It also checks the **/etc/shells** file to verify the login shell for root.

FILES

/etc/ptmp
/etc/shells

SEE ALSO

passwd(1), **vi**(1), **passwd**(5), **adduser**(8)

NAME

vmstat – report virtual memory statistics

SYNOPSIS

vmstat [**–cfisS**] [*interval* [*count*]]

DESCRIPTION

vmstat delves into the system and normally reports certain statistics kept about process, virtual memory, disk, trap and CPU activity.

Without options, **vmstat** displays a one-line summary of the virtual memory activity since the system has been booted. If *interval* is specified, **vmstat** summarizes activity over the last *interval* seconds. If a *count* is given, the statistics are repeated *count* times.

For example, the following command displays a summary of what the system is doing every five seconds. This is a good choice of printing interval since this is how often some of the statistics are sampled in the system.

```
example% vmstat 5

procs   memory              page    faults
r b w  avm  fre  re at  pi  po  fr  de  sr x0 x1 x2 x3  in  sy  cs  us sy  id
2 0 0  918  286  0 0   0   0   0   0   0 1  0  0  0   4  12   5   3  5 91
1 0 0  846  254  0 0   0   0   0   0   0 6  0  1  0  42 153  31   7 40 54
1 0 0  840  268  0 0   0   0   0   0   0 5  0  0  0  27 103  25   8 26 66
1 0 0  620  312  0 0   0   0   0   0   0 6  0  0  0  26  76  25   6 27 67

CTRL-C
example%
```

The fields of **vmstat**'s display are:

procs Report the number of processes in each of the three following states:
- **r** in run queue
- **b** blocked for resources (i/o, paging, etc.)
- **w** runnable or short sleeper (< 20 secs) but swapped

memory Report on usage of virtual and real memory. Virtual memory is considered active if it belongs to processes which are running or have run in the last 20 seconds.
- **avm** number of active virtual Kbytes
- **fre** size of the free list in Kbytes

page Report information about page faults and paging activity. The information on each of the following activities is averaged each five seconds, and given in units per second.
- **re** page reclaims — but see the **–S** option for how this field is modified.
- **at** number of attaches — but see the **–S** option for how this field is modified.
- **pi** kilobytes per second paged in
- **po** kilobytes per second paged out
- **fr** kilobytes freed per second
- **de** anticipated short term memory shortfall in Kbytes
- **sr** pages scanned by clock algorithm, per-second

disk Report number of disk operations per second (this field is system dependent). For Sun systems, four slots are available for up to four drives: "x0" (or "s0" for SCSI disks), "x1", "x2", and "x3".

faults Report trap/interrupt rate averages per second over last 5 seconds.
- **in** (non clock) device interrupts per second
- **sy** system calls per second
- **cs** CPU context switch rate (switches/sec)

cpu Give a breakdown of percentage usage of CPU time.
- **us** user time for normal and low priority processes
- **sy** system time
- **id** CPU idle

OPTIONS

–c Report cache flushing statistics. By default, report the total number of each kind of cache flushed since boot time. The types are: user, context, region, segment, page, and partial-page.

–f Report on the number of **forks** and **vforks** since system startup and the number of pages of virtual memory involved in each kind of fork.

–i Report the number of interrupts per device. Autovectored interrupts (including the clock) are listed first.

–s Display the contents of the **sum** structure, giving the total number of several kinds of paging-related events which have occurred since boot.

–S Report on swapping rather than paging activity. This option will change two fields in **vmstat**'s ''paging'' display: rather than the ''re'' and ''at'' fields, **vmstat** will report ''si'' (swap-ins), and ''so'' (swap-outs).

FILES

/dev/kmem
/vmunix

BUGS

If more than one autovectored device has the same name, interrupts are counted for all like-named devices regardless of unit number. Such devices are listed with a unit number of '**?**'.

NAME

ypbatchupd – NIS batch update daemon

SYNOPSIS

/usr/etc/rpc.ypbatchupd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

ypbatchupd(8C) is the remote service used by **snap**(1) and **logintool**(8) to update the Network Interface Service (NIS) database on the master server, and to push all modified NIS maps to NIS servers. It is normally started by **/etc/rc.local**.

SEE ALSO

snap(1), **logintool**(8), **rc**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypinit – build and install NIS database

SYNOPSIS

/usr/etc/yp/ypinit –m

/usr/etc/yp/ypinit –s *master_name*

DESCRIPTION

ypinit sets up a Network Interface Service (NIS) database on an NIS server. It can be used to set up a master or a slave server. You must be the super-user to run it. It asks a few, self-explanatory questions, and reports success or failure to the terminal.

It sets up a master server using the simple model in which that server is master to all maps in the data base. This is the way to bootstrap the NIS system; later if you want you can change the association of maps to masters.

Note: If there are both 3.x and 4.x NIS servers running in the network, the 4.x server should be configured as the master.

All databases are built from scratch, either from information available to the program at runtime, or from the ASCII data base files in **/etc**. These files are listed below under FILES. All such files should be in their "traditional" form, rather than the abbreviated form used on client machines.

An NIS database on a slave server is set up by copying an existing database from a running server. The *master_name* argument should be the hostname of an NIS server (either the master server for all the maps, or a server on which the data base is up-to-date and stable).

Read **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

OPTIONS

–m Indicate that the local host is to be the NIS master.

–s Set up a slave database.

FILES

/etc/passwd
/etc/group
/etc/hosts
/etc/networks
/etc/services
/etc/protocols
/etc/ethers

SEE ALSO

ypfiles(5), **makedbm**(8), **ypmake**(8), **yppush**(8), **ypserv**(8), **ypxfr**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypmake – rebuild NIS database

SYNOPSIS

cd /var/yp ; make [*map*]

DESCRIPTION

The file called **Makefile** in **/var/yp** is used by **make**(1) to build the Network Interface Service (NIS) database. With no arguments, **make** creates **dbm** databases for any NIS maps that are out-of-date, and then executes **yppush**(8) to notify slave databases that there has been a change.

If you supply a *map* on the command line, **make** will update that map only. Typing **make passwd** will create and **yppush** the password database (assuming it is out of date). Likewise, **make hosts** and **make networks** will create and **yppush** the host and network files, **/etc/hosts** and **/etc/networks**.

There are three special variables used by **make**: DIR, which gives the directory of the source files; NOPUSH, which when non-null inhibits doing a **yppush** of the new database files; and DOM, used to construct a domain other than the master's default domain. The default for DIR is **/etc**, and the default for NOPUSH is the null string.

Refer to **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

FILES

/var/yp
/etc/hosts
/etc/networks

SEE ALSO

make(1), **ypfiles**(5), **makedbm**(8), **yppush**(8), **ypserv**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

yppasswdd, rpc.yppasswdd – server for modifying NIS password file

SYNOPSIS

/usr/etc/rpc.yppasswdd *filename* [*adjunct_file*] [**–nogecos**] [**–noshell**] [**–nopw**]
[**–m** *argument1 argument2 ...*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

yppasswdd is a server that handles password change requests from **yppasswd**(1). Unless an *adjunct_file* is specified, it changes a password entry in *filename*, which is assumed to be in the format of **passwd**(5). *filename* is the password file that provides the basis for the *passwd.byname* and *passwd.byuid* maps. This should not be confused with the servers */etc/passwd* file which controls access to the server. In particular this file should not contain an entry for the super user.

If an *adjunct_file* is specified or **/etc/security/passwd.adjunct** exists, this file will be changed instead of the *filename*. An entry in *filename* or *adjunct_file* will only be changed if the password presented by **yppasswd**(1) matches the encrypted password of that entry.

If the **–noshell** **–nogecos** or **–nopw** options are given then these fields may not be changed remotely using **chfn**, **chsh**, or **passwd**(1).

If the **–m** option is given, then after *filename* or *adjunct_file* is modified, a **make**(1) will be performed in */var/yp*. Any arguments following the flag will be passed to *make*.

This server is not run by default, nor can it be started up from **inetd**(8C). If it is desired to enable remote password updating for the Network Interface Service (NIS), then an entry for **yppasswdd** should be put in the **/etc/rc** file of the host serving as the master for the NIS **passwd** file.

EXAMPLE

If the NIS password file is stored as **/var/yp/passwd**, then to have password changes propagated immediately, the server should be invoked as

/usr/etc/rpc.yppasswdd /var/yp/passwd –m passwd DIR=/var/yp

FILES

/var/yp/Makefile
/etc/security/passwd.adjunct
/etc/rc

SEE ALSO

make(1), **yppasswd**(1), **passwd**(1), **passwd**(5), **passwd.adjunct**(5), **ypfiles**(5), **inetd**(8C), **ypmake**(8)

NOTES

The password file specified to *rpc.yppasswdd* may not be a link.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

yppoll – version of NIS map at NIS server

SYNOPSIS

/usr/etc/yp/yppoll [**–h** *host*] [**–d** *domain*] *mapname*

DESCRIPTION

yppoll asks a **ypserv**(8) process what the order number is, and which host is the Network Interface Service (NIS) master server for the named map. If the server is a v.1 NIS protocol server, **yppoll** uses the older protocol to communicate with it. In this case, it also uses the older diagnostic messages in case of failure.

OPTIONS

–h *host* Ask the **ypserv** process at *host* about the map parameters. If *host* is not specified, the NIS server for the local host is used. That is, the default host is the one returned by **ypwhich**(8).

–d *domain*

Use *domain* instead of the default domain.

SEE ALSO

ypfiles(5), **ypserv**(8), **ypwhich**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

yppush – force propagation of changed NIS map

SYNOPSIS

/usr/etc/yp/yppush [**–v**] [**–d** *domain*] *mapname*

DESCRIPTION

yppush copies a new version of a Network Interface Service (NIS) map from the master NIS server to the slave NIS servers. It is normally run only on the master NIS server by the **Makefile** in **/var/yp** after the master databases are changed. It first constructs a list of NIS server hosts by reading the NIS map **ypservers** within the *domain*. Keys within the map **ypservers** are the ASCII names of the machines on which the NIS servers run.

A "transfer map" request is sent to the NIS server at each host, along with the information needed by the transfer agent (the program which actually moves the map) to call back the **yppush** . When the attempt has completed (successfully or not), and the transfer agent has sent **yppush** a status message, the results may be printed to stdout. Messages are also printed when a transfer is not possible; for instance when the request message is undeliverable, or when the timeout period on responses has expired.

Refer to **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

OPTIONS

–d *domain*

Specify a *domain*.

–v Verbose. This prints messages when each server is called, and for each response. If this flag is omitted, only error messages are printed.

FILES

/var/yp/*domain***/ypservers.{***dir***,***pag***}**
/var/yp

SEE ALSO

ypfiles(5), **ypserv**(8), **ypxfr**(8)

NIS protocol specification

BUGS

In the current implementation (version 2 NIS protocol), the transfer agent is **ypxfr**(8), which is started by the **ypserv** program. If **yppush** detects that it is speaking to a version 1 NIS protocol server, it uses the older protocol, sending a version 1 YPPROC_GET request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer is performed for version 1 servers. **yppush** prints a message saying that an "old-style" message has been sent. The system administrator should later check to see that the transfer has actually taken place.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypserv, ypbind, ypxfrd – NIS server and binder processes

SYNOPSIS

/usr/etc/ypserv [–d]

/usr/etc/ypbind [–s] [–ypset | –ypsetme]

ypxfrd [–x]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

The Network Interface Service (NIS) provides a simple network lookup service consisting of databases and processes. The databases are **dbm**(3X) files in a directory tree rooted at **/var/yp**. These files are described in **ypfiles**(5). The processes are **/usr/etc/ypserv**, the NIS database lookup server, and **/usr/etc/ypbind**, the NIS binder. The programmatic interface to the NIS service is described in **ypclnt**(3N). Administrative tools are described in **yppush**(8), **ypxfr**(8), **yppoll**(8), **ypwhich**(8), and **ypset**(8). Tools to see the contents of NIS maps are described in **ypcat**(1), and **ypmatch**(1). Database generation and maintenance tools are described in **ypinit**(8), **ypmake**(8), and **makedbm**(8).

Both **ypserv** and **ypbind** are daemon processes typically activated at system startup time from **/etc/rc.local**. **ypserv** runs only on NIS server machines with a complete NIS database. **ypbind** runs on all machines using the NIS services, both NIS servers and clients.

ypxfrd transfers entire NIS maps in an efficient manner. For systems that use this daemon, map transfers will be 10 to 100 times faster, depending on the map. To use this daemon, **ypxfrd** should be run on a server running SunOS release 4.1. **ypxfr** will attempt to use **ypxfrd** first, if that fails, it will print a warning and then use the older transfer method.

The **ypserv** daemon's primary function is to look up information in its local database of NIS maps. The operations performed by **ypserv** are defined for the implementor by the *YP Protocol Specification*, and for the programmer by the header file **rpcsvc/yp_prot.h**. Communication to and from **ypserv** is by means of RPC calls. Lookup functions are described in **ypclnt**(3N), and are supplied as C-callable functions in the C library. There are four lookup functions, all of which are performed on a specified map within some NIS domain: **match**, **get_first**, **get_next**, and **get_all**. The **match** operation takes a key, and returns the associated value. The **get_first** operation returns the first key-value pair from the map, and **get_next** can be used to enumerate the remainder. **get_all** ships the entire map to the requester as the response to a single RPC request.

Two other functions supply information about the map, rather than map entries: **get_order_number**, and **get_master_name**. In fact, both order number and master name exist in the map as key-value pairs, but the server will not return either through the normal lookup functions. If you examine the map with **makedbm**(8), however, they will be visible. Other functions are used within the NIS service subsystem itself, and are not of general interest to NIS clients. They include **do_you_serve_this_domain?**, **transfer_map**, and **reinitialize_internal_state**.

The function of **ypbind** is to remember information that lets client processes on a single node communicate with some **ypserv** process. **ypbind** must run on every machine which has NIS client processes; **ypserv** may or may not be running on the same node, but must be running somewhere on the network.

The information **ypbind** remembers is called a *binding* — the association of a domain name with the internet address of the NIS server, and the port on that host at which the **ypserv** process is listening for service requests. This information is cached in the directory **/var/yp/binding** using a filename of **domainname.version**.

The process of binding is driven by client requests. As a request for an unbound domain comes in, the **ypbind** process broadcasts on the net trying to find a **ypserv** process that serves maps within that domain. Since the binding is established by broadcasting, there must be at least one **ypserv** process on every net. If

the client is running in C2 secure mode, then **ypbind** will only accept bindings to servers where the **ypserv** process is running as root. Once a domain is bound by a particular **ypbind**, that same binding is given to every client process on the node. The **ypbind** process on the local node or a remote node may be queried for the binding of a particular domain by using the **ypwhich**(1) command.

Bindings and rebindings are handled transparently by the C library routines. If **ypbind** is unable to speak to the **ypserv** process it's bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind the domain once again. Requests received for an unbound domain will wait until the domain requested is bound. In general, a bound domain is marked as unbound when the node running **ypserv** crashes or gets overloaded. In such a case, **ypbind** will to bind any NIS server (typically one that is less-heavily loaded) available on the net.

ypbind also accepts requests to set its binding for a particular domain. The request is usually generated by the NIS subsystem itself. **ypset**(8) is a command to access the **set_domain** facility. It is for unsnarling messes. Note: the **set_domain** procedure only accepts requests from processes running as root.

OPTIONS

–d The NIS service should go to the DNS (Domain Name Service) for more host information.

–s Secure. When specified, only ypservers bound to a reserved port are used. This allows for a slight increase in security in completely controlled environments, where there are no computers operated by untrusted individuals. It offers no real increase in security.

–v Do not fork when **ypxfrd** is called multiple times.

–ypset **ypset**(8) may be used to change the binding. This option is very dangerous, and only should be used for debugging the network from a remote machine.

–ypsetme

ypset(8) may be issued from this machine, security is based on IP address checking, which can be defeated on network where untrusted individuals may inject packets. This option is not recommended.

FILES

If the file **/var/yp/ypserv.log** exists when **ypserv** starts up, log information will be written to this file when error conditions arise.

The file(s) **/var/yp/binding/domainname.version** will be created to speed up the binding process. These files cache the last successful binding created for the given domain, when a binding is requested these files are checked for validity and then used.
/var/yp
/usr/etc/ypbind

SEE ALSO

domainname(1), **ypcat**(1), **ypmatch**(1), **dbm**(3X), **ypclnt**(3N), **ypfiles**(5) **makedbm**(8), **ypmake**(8), **ypinit**(8), **yppoll**(8), **yppush**(8), **ypset**(8), **ypwhich**(8), **ypxfr**(8),

Network Programming
System and Network Administration

NOTES

Both **ypbind** and **ypserv** support multiple domains. The **ypserv** process determines the domains it serves by looking for directories of the same name in the directory **/var/yp**. It will reply to all broadcasts requesting yp service for that domain. Additionally, the **ypbind** process can maintain bindings to several domains and their servers, the default domain is however the one specified by the **domainname**(1) command at startup time.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypset – point ypbind at a particular server

SYNOPSIS

/usr/etc/yp/ypset [**–V1**|**–V2**] [**–d** *domain*] [**–h** *host*] *server*

DESCRIPTION

ypset tells **ypbind** to get the Network Interface Service (NIS) for the specified *domain* from the **ypserv** process running on *server*. If *server* is down, or is not running **ypserv**, this is not discovered until an NIS client process tries to get a binding for the domain. At this point, the binding set by **ypset** is tested by **ypbind**. If the binding is invalid, **ypbind** attempts to rebind for the same domain.

ypset is useful for binding a client node which is not on a broadcast net, or is on a broadcast net which is not running an NIS server host. It also is useful for debugging NIS client applications, for instance where an NIS map only exists at a single NIS server host.

In cases where several hosts on the local net are supplying NIS services, it is possible for **ypbind** to rebind to another host even while you attempt to find out if the **ypset** operation succeeded. For example, you can type:

```
example% ypset host1
example% ypwhich
host2
```

which can be confusing. This is a function of the NIS service subsystem's attempt to load-balance among the available NIS servers, and occurs when *host1* does not respond to **ypbind** because it is not running **ypserv** (or is overloaded), and *host2*, running **ypserv**, gets the binding.

server indicates the NIS server to bind to, and can be specified as a name or an IP address. If specified as a name, **ypset** attempts to use NIS services to resolve the name to an IP address. This works only if the node has a current valid binding for the domain in question. In most cases, *server* should be specified as an IP address.

Refer to **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

OPTIONS

–V1 Bind *server* for the (old) v.1 NIS protocol.

–V2 Bind *server* for the (current) v.2 NIS protocol.

If no version is supplied, **ypset**, first attempts to set the domain for the (current) v.2 protocol. If this attempt fails, **ypset**, then attempts to set the domain for the (old) v.1 protocol.

–h*host* Set ypbind's binding on *host*, instead of locally. *host* can be specified as a name or as an IP address.

–d*domain*
Use *domain* , instead of the default domain.

DIAGNOSTICS

Sorry, I couldn't send my rpc message to ypbind on host *name*
The user is not root, or **ypbind** was run without one of the **–ypset** flags. See **ypserv**(8) for explanations of the **–ypset** flags.

SEE ALSO

ypwhich(1), **ypfiles**(5), **ypserv**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypsync – collect most up-to-date NIS maps

SYNOPSIS

/usr/etc/yp/ypsync [**–r**] [**–u**]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0*.x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

ypsync gathers current Network Information Service (NIS) maps to the local NIS server. When invoked with no arguments, it polls all the NIS servers listed in the **/etc/ypservers** NIS map for the maps they serve, and the order of those maps. If there are any new maps that the local server does not have, or if there are maps that are more current than the local server's copy, it excutes **ypxfr**(8) to transfer those maps to the local server.

ypsync eliminates the need for **cron**(8) jobs to ensure that NIS map updates are eventually transmitted to all NIS servers, and supports different NIS maps having different masters. It is invoked periodically by **ypserv**(8).

OPTIONS

–r When invoked with the **–r** flag, **ypsync** re-creates the local **/var/yp** directory and databases if needed. This facility is used when upgrading servers, since they can automatically retrieve NIS maps without needing manual intervention. The NIS master of the **ypservers** map can also designate new servers, which would automatically pick up their new maps on reboot.

–u When invoked with the **–u** flag, **ypsync** updates the list of NIS servers on the master of the **ypservers** NIS map to include the local system if it does not already, and then get copies of all the NIS databases. A user invoking **ypsync –u** may not be root, and must have the *networks privilege in the* NIS **group** map.

FILES

/var/yp/YP.*domainname*

SEE ALSO

ypupdate(3), **ypserv**(8), **ypxfr**(8)

Sun386i Advanced Administration

System and Network Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypupdated, rpc.ypupdated – server for changing NIS information

SYNOPSIS

rpc.ypupdated [**–is**]

DESCRIPTION

ypupdated is a daemon that updates information in the Network Interface Service (NIS), normally started up by **inetd**(8C). **ypupdated** consults the file **updaters**(5) in the directory **/var/yp** to determine which NIS maps should be updated and how to change them.

By default, the daemon requires the most secure method of authentication available to it, either DES (secure) or UNIX (insecure).

OPTIONS

–i Accept RPC calls with the insecure AUTH_UNIX credentials. This allows programmatic updating of the NIS maps in all networks.

–s Accept only calls authenticated using the secure RPC mechanism (AUTH_DES authentication). This disables programmatic updating of the NIS maps unless the network supports these calls.

FILES

/var/yp/updaters

SEE ALSO

updaters(5), **inetd**(8C), **keyserv**(8C)

System and Network Administration
Network Programming

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypxfr – transfer NIS map from NIS server to here

SYNOPSIS

/usr/etc/yp/ypxfr [**–b**] [**–c**] [**–f**] [**–d** *domain*] [**–h** *host*] [**–s** *domain*] [**–C** *tid prog ipadd port*] *mapname*

DESCRIPTION

ypxfr moves a Network Interface Service (NIS) map in the default domain for the local host to the local host by making use of normal NIS services. It creates a temporary map in the directory **/var/yp/***domain* (this directory must already exist; *domain* is the default domain for the local host), fills it by enumerating the map's entries, fetches the map parameters (master and order number), and loads them. It then deletes any old versions of the map and moves the temporary map to the real *mapname*.

If run interactively, **ypxfr** writes its output to the terminal. However, if it is invoked without a controlling terminal, and if the log file **/var/yp/ypxfr.log** exists, it will append all its output to that file. Since **ypxfr** is most often run from the super-user's **crontab** file, or by **ypserv**, you can use the log file to retain a record of what was attempted, and what the results were.

If **issecure**(3) is TRUE, **ypxfr** requires that **ypserv** on the *host* be running as root. If the map being transferred is a secure map, **ypxfr** sets the permissions on the map to 0600.

For consistency between servers, **ypxfr** should be run periodically for every map in the NIS data base. Different maps change at different rates: the *services.byname* map may not change for months at a time, for instance, and may therefore be checked only once a day (in the wee hours). You may know that *mail.aliases* or *hosts.byname* changes several times per day. In such a case, you may want to check hourly for updates. A **crontab**(5) entry can be used to perform periodic updates automatically. Rather than having a separate **crontab** entry for each map, you can group commands to update several maps in a shell script. Examples (mnemonically named) are in **/usr/etc/yp**: **ypxfr_1perday**, **ypxfr_2perday**, and **ypxfr_1perhour**. They can serve as reasonable first cuts.

Refer to **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

OPTIONS

–b Preserve the resolver flag in the map during the transfer.

–c Do not send a "Clear current map" request to the local **ypserv** process. Use this flag if **ypserv** is not running locally at the time you are running **ypxfr**. Otherwise, **ypxfr** will complain that it cannot talk to the local **ypserv**, and the transfer will fail.

–f Force the transfer to occur even if the version at the master is not more recent than the local version.

–d *domain*
Specify a domain other than the default domain.

–h *host* Get the map from *host*, regardless of what the map says the master is. If *host* is not specified, **ypxfr** asks the NIS service for the name of the master, and tries to get the map from there. *host* may be a name or an internet address in the form *a.b.c.d*.

–s *domain*
Specify a source domain from which to transfer a map that should be the same across domains (such as the *services.byname* map).

–C*tid prog ipadd port*
This option is **only** for use by **ypserv**. When **ypserv** invokes **ypxfr**, it specifies that **ypxfr** should call back a **yppush** process at the host with IP address *ipaddr*, registered as program number *prog*, listening on port *port*, and waiting for a response to transaction *tid*.

FILES

/var/yp/ypxfr.log log file
/usr/etc/yp/ypxfr_1perday
script to run one transfer per day, for use with **cron**(8)
/usr/etc/yp/ypxfr_2perday
script to run two transfers per day
/usr/etc/yp/ypxfr_1perhour
script for hourly transfers of volatile maps
/var/yp/*domain* NIS domain
/var/spool/cron/crontabs/root
Super-user's **crontab** file

SEE ALSO

issecure(3), **crontab**(5), **ypfiles**(5), **cron**(8), **ypserv**(8), **yppush**(8)

YP Protocol Specification, in *Network Programming*

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

zdump – time zone dumper

SYNOPSIS

zdump [**–v**] [**–c** *cutoffyear*] [*zonename* ...]

DESCRIPTION

zdump prints the current time in each *zonename* named on the command line.

OPTIONS

–v For each *zonename* on the command line, print the current time, the time at the lowest possible time value, the time one day after the lowest possible time value, the times both one second before and exactly at each time at which the rules for computing local time change, the time at the highest possible time value, and the time at one day less than the highest possible time value. Each line ends with **isdst=1** if the given time is Daylight Saving Time or **isdst=0** otherwise.

–c *cutoffyear*

Cut off the verbose output near the start of the year *cutoffyear*.

FILES

/usr/share/lib/zoneinfo standard zone information directory

SEE ALSO

ctime(3V), **tzfile**(5), **zic**(8)

NAME

zic – time zone compiler

SYNOPSIS

zic [**–v**] [**–d** *directory*] [**–l** *localtime*] [*filename* ...]

DESCRIPTION

zic reads text from the file(s) named on the command line and creates the time conversion information files specified in this input. If a *filename* is '–', the standard input is read.

Input lines are made up of fields. Fields are separated from one another by any number of white space characters. Leading and trailing white space on input lines is ignored. An '#' (unquoted sharp character) in the input introduces a comment which extends to the end of the line the sharp character appears on. White space characters and sharp characters may be enclosed in ' "' (double quotes) if they're to be used as part of a field. Any line that is blank (after comment stripping) is ignored. Non-blank lines are expected to be of one of three types: rule lines, zone lines, and link lines.

A rule line has the form

Rule NAME FROM TO TYPE IN ON AT SAVE LETTER/S

For example:

Rule USA 1969 1973 – Apr lastSun 2:00 1:00 D

The fields that make up a rule line are:

NAME Gives the (arbitrary) name of the set of rules this rule is part of.

FROM Gives the first year in which the rule applies. The word **minimum** (or an abbreviation) means the minimum year with a representable time value. The word **maximum** (or an abbreviation) means the maximum year with a representable time value.

TO Gives the final year in which the rule applies. In addition to **minimum** and **maximum** (as above), the word **only** (or an abbreviation) may be used to repeat the value of the **FROM** field.

TYPE Gives the type of year in which the rule applies. If **TYPE** is '–' then the rule applies in all years between **FROM** and **TO** inclusive; if **TYPE** is **uspres**, the rule applies in U.S. Presidential election years; if **TYPE** is **nonpres**, the rule applies in years other than U.S. Presidential election years. If **TYPE** is something else, then **zic** executes the command

yearistype *year type*

to check the type of a year: an exit status of zero is taken to mean that the year is of the given type; an exit status of one is taken to mean that the year is not of the given type.

IN Names the month in which the rule takes effect. Month names may be abbreviated.

ON Gives the day on which the rule takes effect. Recognized forms include:

5	the fifth of the month
lastSun	the last Sunday in the month
lastMon	the last Monday in the month
Sun>=8	first Sunday on or after the eighth
Sun<=25	last Sunday on or before the 25th

Names of days of the week may be abbreviated or spelled out in full. Note: there must be no spaces within the **ON** field.

AT Gives the time of day at which the rule takes effect. Recognized forms include:

2	time in hours
2:00	time in hours and minutes
15:00	24-hour format time (for times after noon)
1:28:14	time in hours, minutes, and seconds

Any of these forms may be followed by the letter **w** if the given time is local ''wall clock'' time or **s** if the given time is local ''standard'' time; in the absence of **w** or **s**, wall clock time is assumed.

SAVE Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the **AT** field (although, of course, the **w** and **s** suffixes are not used).

LETTER/S
Gives the ''variable part'' (for example, the ''S'' or ''D'' in ''EST'' or ''EDT'') of time zone abbreviations to be used when this rule is in effect. If this field is '–', the variable part is null.

A zone line has the form

Zone NAME GMTOFF RULES/SAVE FORMAT [UNTIL]

For example:

Zone Australia/South–west 9:30 Aus CST 1987 Mar 15 2:00

The fields that make up a zone line are:

NAME The name of the time zone. This is the name used in creating the time conversion information file for the zone.

GMTOFF
The amount of time to add to GMT to get standard time in this zone. This field has the same format as the **AT** and **SAVE** fields of rule lines; begin the field with a minus sign if time must be subtracted from GMT.

RULES/SAVE
The name of the rule(s) that apply in the time zone or, alternately, an amount of time to add to local standard time. If this field is '–' then standard time always applies in the time zone.

FORMAT
The format for time zone abbreviations in this time zone. The pair of characters **%s** is used to show where the ''variable part'' of the time zone abbreviation goes. **UNTIL** The time at which the GMT offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. If this is specified, the time zone information is generated from the given GMT offset and rule change until the time specified.

The next line must be a ''continuation'' line; this has the same form as a zone line except that the string ''Zone'' and the name are omitted, as the continuation line will place information starting at the time specified as the **UNTIL** field in the previous line in the file used by the previous line. Continuation lines may contain an **UNTIL** field, just as zone lines do, indicating that the next line is a further continuation.

A link line has the form

Link LINK-FROM LINK-TO

For example:

Link US/Eastern EST5EDT

The **LINK-FROM** field should appear as the **NAME** field in some zone line; the **LINK-TO** field is used as an alternate name for that zone.

Except for continuation lines, lines may appear in any order in the input.

OPTIONS

–v Complain if a year that appears in a data file is outside the range of years representable by system time values (0:00:00 AM GMT, January 1, 1970, to 3:14:07 AM GMT, January 19, 2038).

–d *directory* Create time conversion information files in the directory **directory** rather than in the standard directory **/usr/share/lib/zoneinfo.**

–l *timezone* Use the time zone *timezone* as local time. **zic** will act as if the file contained a link line of the form

Link *timezone* **localtime**

FILES

/usr/share/lib/zoneinfo standard directory used for created files

SEE ALSO

time(1V), **ctime**(3V), **tzfile**(5), **zdump**(8)

NOTES

For areas with more than two types of local time, you may need to use local standard time in the **AT** field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z