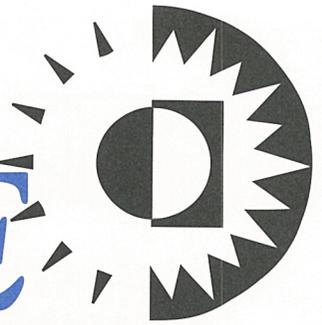


INSIDE SOLARIS™

Tips & techniques for users of SunSoft Solaris



IN THIS ISSUE

- 1**
StarOffice

- 5**
Who manages the Internet?

- 9**
Setting up virtual terminals

- 10**
Automating account creation

- 15**
Solaris Q&A



High-Quality Content Prevails!

StarOffice

by Rainer Dorsch

Since November 1997, SUN Microsystems bundles StarOffice of StarDivision (Germany) with each shipped Ultra Workstation. StarOffice 4.0, including

- a text processor
- a spread sheet
- a presentation program
- a database
- an Internet browser
- an E-mail program
- a news reader
- a file manager

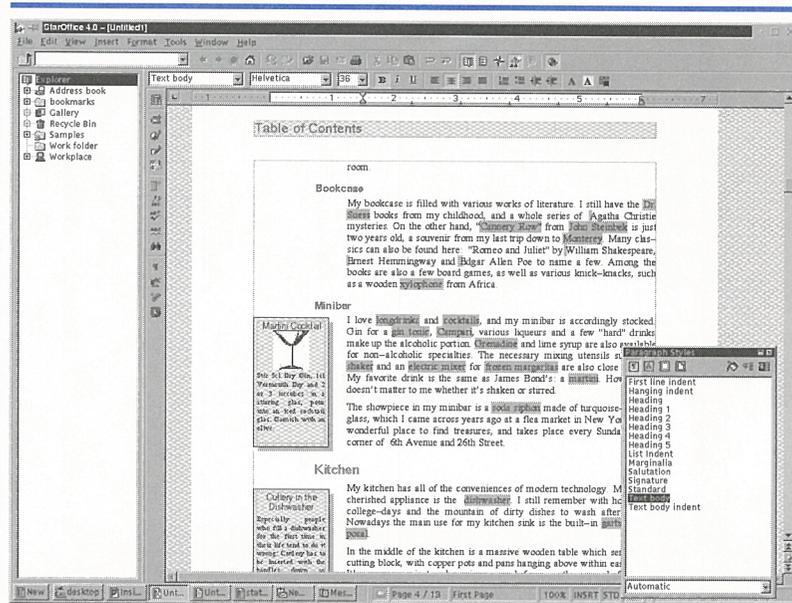
StarOffice is an office suite comparable to Microsoft Office. This

equivalence is not limited to the functionality, but the appearance (menus and icons) is similar, too. Also, network integration is superior.

The Chameleon

StarOffice realizes the *Do Everything in One Place* philosophy: in one application, all features are unified. Changing from one document to another with the task bar (bottom row in the screenshots), changes the outfit of the program from a text processor (Figure A), to a spreadsheet (Figure B on page 2), to a presentation program (Figure C on

Figure A



StarOffice uses a unified interface.

THE COBB GROUP IS NOW ZD JOURNALS!

Dear Subscriber:

In 1991, The Cobb Group became a part of **Ziff-Davis Inc.** Our affiliation with technology's leading integrated media and marketing company allowed us to achieve higher quality standards and expand our delivery mediums—ensuring that all our customers get what they need, when they want it.

Today, as **ZD Journals** (a part of ZD Education) we further strengthen our promise to provide you with the same insightful, trustworthy information and proven techniques on the latest products and technologies.

Thanks to you, we are the leading source of "how-to" publications for computer users—we appreciate your business and look forward to providing you with more great information through the best medium available.

Sincerely, 

G. A. Weissberg
Executive VP and General Manager
ZD Journals



page 3), to an Internet browser and Web editor (**Figure D**), to an E-mail program (**Figure E** on page 4), or to a news reader (**Figure F** also on page 4).

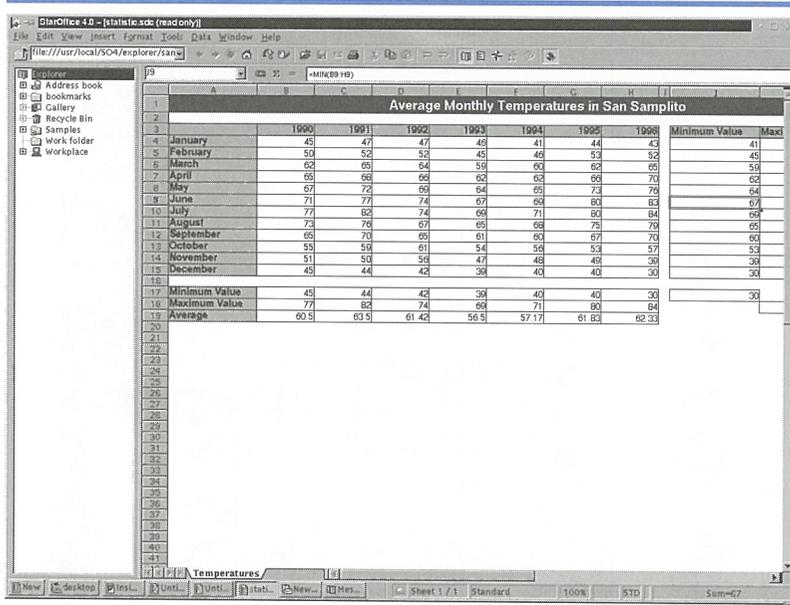
Why or why not StarOffice?

StarOffice is available for many platforms, including Solaris, Windows 95/NT, OS/2, Linux, and MacOS. Therefore, StarOffice is a good solution for a heterogeneous hardware environment. On all platforms, StarOffice has the same functionality, so exchanging documents between different platforms doesn't cause any problems. In other words, there's no need for Solaris users to switch to Windows 95/NT to prepare a technical report or a presentation. One platform per user enhances overall productivity.

The Internet integration of StarOffice is excellent. You can send a document by E-mail to a colleague (who may be working on a different platform) in seconds. Exporting to HTML format is supported by all applications. You can also store files transparently on a remote FTP or Web server.

StarOffice for UNIX still looks like an MS Windows application. This is disappointing for experienced UNIX users who expect the EMACS key bindings to work and an X11-like mouse behavior for cut

Figure B



View a spreadsheet in StarOffice.

and paste. This is a considerable advantage for an experienced Windows 95/MS Office user, such as a secretary, who can start immediately on a UNIX workstation without having UNIX knowledge. If the number of platforms can be reduced this way, administration costs are cut.

Many people use Solaris at work and Linux at home. Since the Linux version of StarOffice is free for personal use, these users can install the same office application at work and at home. This increases the acceptance of StarOffice and the productivity of the user considerably.

Although StarOffice 4.0 can read and write Microsoft Office 95 formats, the filters don't work for many documents. As a rule of thumb, the filters will work for everything that can be converted to RTF, and other formatting will be destroyed. If you receive many Microsoft Office documents, and RTF level exchange isn't enough, StarOffice 4.0 can't be used. StarDivision claims that StarOffice 5.0 (announced for the second quarter of 1998) will contain much more sophisticated filters for Microsoft Office97.

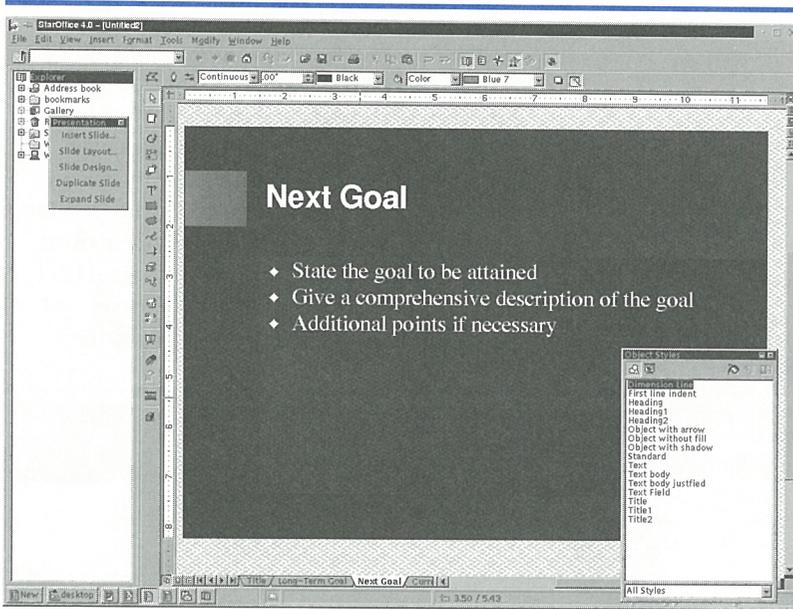
Installation

In this section, we'll describe the installation process of StarOffice 4.0 Service Pack 2 (SP 2) and comment on SP 3. The Service Pack number can be interpreted as the next digit in the release number. SP 3 is, at this time, only available for Linux, but Solaris and Windows 95/NT releases have been announced.

There are two installation procedures: a single user installation and a network installation (not available before SP 2). The single user installation allows each user to install the complete system in his home directory without root permissions. A network installation installs StarOffice in a global directory and each user has user modifiable files in his home directory. Since a network installation is appropriate for Solaris, we'll discuss this method in detail.

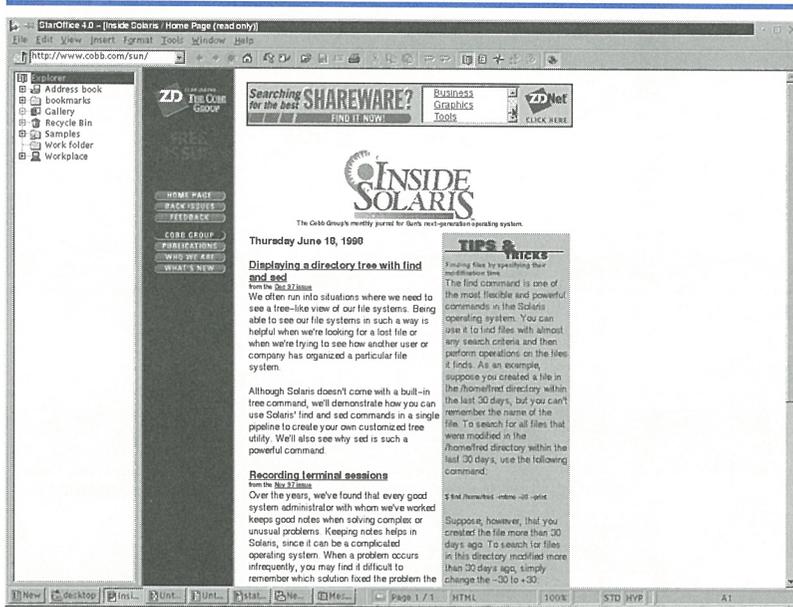
The Solaris CD-ROM contains directories for different languages. (SP 2 offers English, German, French, Dutch, and Swedish). The directories differ in the language of the menus, online help, and the available dictionaries for spell-checking. The menu language seems to be hard-coded into the binary. All directories contain an English dictionary and a language-specific dictionary.

Figure C



This shows the presentation program in StarOffice.

Figure D



Use StarOffice's Web browser and editor.

After mounting the CD-ROM (assume at /cdrom), to start the installation, enter

```
/cdrom/english/prod_sols/setup /net
```

for the English language version. This starts the graphical utility for installation and de-installation. If an administrator wants to install the English dictionary (in addition to the menu language dictionary), *Custom Installation* should be chosen; otherwise, *Standard*

Installation is sufficient. After the specification of an installation directory (which should be on a partition with about 120MB of free space), the installation starts. The following steps, assume that `/opt/Office40` has been chosen as the installation directory.

After leaving the setup utility, the system administrator has to make the printer queues known to StarOffice. This is done by editing `/opt/Office40/xp3/Xpdefaults`. To add a Postscript printer queue (here `hp3` printing on an HP III Postscript printer)

and a preview device, add the following lines in the sections `[devices]` and `[ports]`:

```
[devices]
    HP LaserJet III PostScript=HP111522
PostScript, hp5a
    PreViewDevice=GENERIC
PostScript, PreViewPort
```

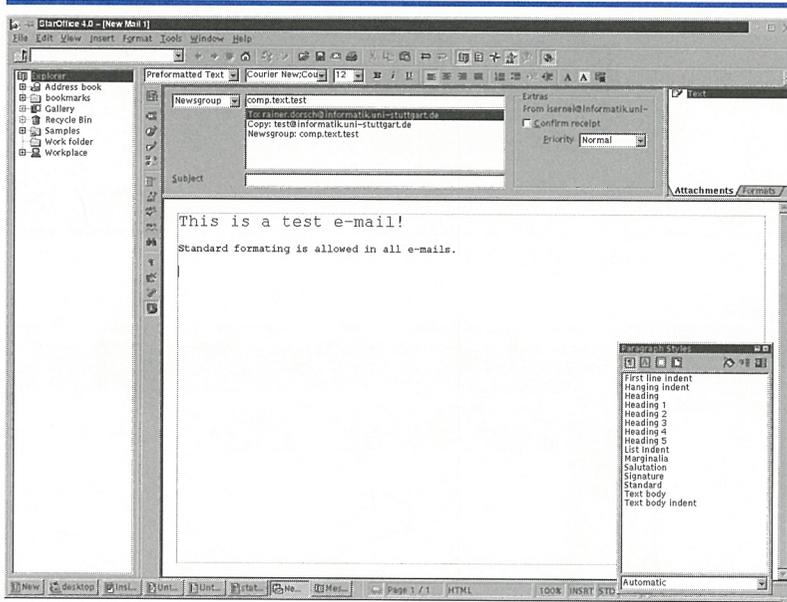
```
[ports]
    PreViewPort=/usr/bin/X11/gv - &
    hp5a=lp -d hp5a
```

In this example, *HP LaserJet III PostScript* and *PreViewDevice* page will appear in the printing dialog of StarOffice. `hp5a` and `PreViewPort` are symbolic names which link a shell command to an entry in the printing dialog. *HP111522 PostScript* and *GENERIC PostScript* are printer definitions. The corresponding definitions can be found in `/opt/Office40/xp3/ppds/GENERIC.ps` and `/opt/Office40/xp3/ppds/HP111522.ps`. A complete list of available printer definitions is listed in the `[other-devices]` section. Margins, paper size, scale, and number of copies can be changed in sections `[GENERIC, PostScript, PreViewPort]` and `[HP111522 PostScript, hp5a]` respectively. The default printer is in the windows section in the same format as alternative printers in the `devices` section. In Service Pack 3, the configuration of the printers can be done alternatively with the graphical printer setup utility `/opt/Office40/bin/psetup`.

After this, each user has to do a personalization of StarOffice. This is done by calling `/opt/Office40/bin/setup`. Here, it is important that the users do not select *Standard Installation* or *Custom Installation*, but *Installation from Net or CD!* Otherwise, the complete software is copied to the user's home directory, not only the user modifiable files. Each user has to enter her personal data, such as name, institution, telephone, fax, and E-mail, which are used in documents created by the auto-pilot. These data can be changed or completed in StarOffice (Tools->Options->General->User Data). Each user requires about 10MB of data in their home directory (about 2MB in SP 3).

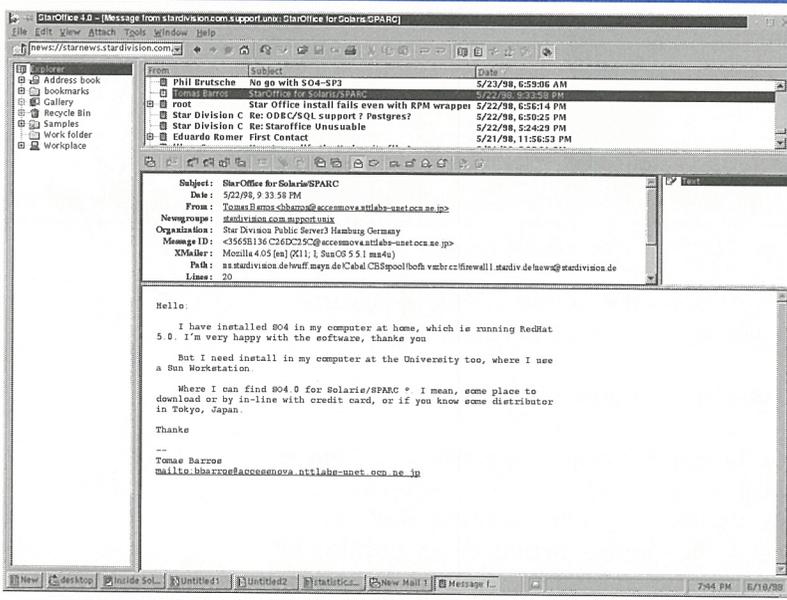
After personalization, StarOffice can be started with *soffice*, if `/opt/Office40/bin` is added to the `PATH` environment variable. One of the first things most users want to do is to change the working directory to the home directory. This is done in Tools/Options/General/Paths/Working Directory.

Figure E



This is the E-mail reader that comes with StarOffice.

Figure F



Here's a view of StarOffice's NNTP news reader.

Performance and stability

In StarOffice 3.1 for UNIX, performance and stability were major problems. This has changed in StarOffice 4.0. The StarDivision developers removed Motif completely and replaced it with their own graphical library, which is considerably faster.

The stability improves with each Service Pack. For SP 2, you can expect one crash per day as a rule of thumb (and less if it's only running in the background). The data at crash time can usually be restored after restart, even if not explicitly saved. SP 3 improves the stability further.

Documentation

The OEM version comes only with online documentation. Currently there is no

printed English documentation for StarOffice. For problems that you can't solve with the online documentation, you must use StarDivision's support channels (including supported news groups). There is a German manual for StarOffice and much German third-party documentation. It is expected that printed English documentation will appear soon on the market.

Further sources of information and references

StarDivision Homepage:

www.stardivision.com

A non-official FAQ for the UNIX platform:

www.on-line.de/~michael.hoennig/soffice4-linux-faq-01.html. ❖

Rainer Dorsch received his masters in Physics in 1996 at the University of Ulm and works now at the University of Stuttgart on his PhD in Computer Science. He started with UNIX in 1992, worked with Linux at home since 1994 and has been heavily involved in Solaris administration since 1996. You may reach him at rainer.dorsch@informatik.uni-stuttgart.de.

The power of whois

Who manages the Internet?

by Lance Spitzner

Who manages the Internet? There are a lot of issues within these basic questions: Who controls IP addresses? Who assigns domain names? Who handles the domain name resolution? This article will answer these questions with a basic overview of how the Internet works and what organizations manage it. We'll also present a basic overview of how the Internet is currently managed and how you can use this knowledge with the command *whois*. We advise you that due to the dynamic nature of the Internet, this information can change from the time we wrote the article to the time it was printed.

There are many critical resources that must be managed for the Internet. Two resources that we'll be focusing on are the management of IP addressing and domain names. IP addresses are unique numbers; each address consists of four octets (32 bits), as specified in RFC 791. Domain names are the organization and representation of IP addresses. In the first part of this

article, we'll discuss IP addressing and how the Internet manages it. We'll then cover the far more complicated and political issue of domain names and how they're controlled.

IP addressing

IP addresses are the workhorses of the Internet—how your packet gets from point A to point B. IPs work because no two addresses are the same. Without a standardized system of unique addressing, the Internet couldn't function. But who is in charge of IP addresses? How do you know that the IP address you have is truly unique? The place to start is the Internet Assigned Numbers Authority (IANA) (www.isi.edu/div7/iana/).

IANA, located at the Information Sciences Institute at the University of Southern California, is responsible for a variety of Internet issues, including IP addressing and domain registration for countries (which we'll discuss later). IANA is the ultimate source of authority for IP addresses and is

ultimately responsible for most of the IP addresses in the world.

IANA controls the IP addresses in a hierarchical manner. First, the organization distributes IP addresses in large blocks to three regional registries. Each block is unique and separate from the other two. Then, each regional registry distributes these IP blocks into smaller blocks to ISPs or large organizations within their region. These ISPs, in turn, distribute IP addresses to smaller ISPs, companies, schools, etc. Each organization manages the IP distribution to the next lower level—ensuring IP addresses aren't wasted or replicated.

The three main regional IP registries are as follows (note that all three registries are not-for-profit organizations):

- RIPE (www.ripe.net) is the Reseaux IP Europeans (more commonly called the Regional Internet Registry for Europe). Located in Amsterdam, RIPE provides support to approximately 1000 Internet Registries, or ISPs, located in Europe, the Middle East, and parts of Asia and Africa (check www.ripe.net/centr/tld.html to see all the countries).
- APNIC is the Asian Pacific Network Information Center. Located in Tokyo, Japan, APNIC provides support for all Asian countries. Currently there isn't a list of every individual country that falls under APNIC. Visit their Web site at www.apnic.net.
- ARIN (www.arin.net) is the American Registry for Internet Numbers. Located in Chantilly, Virginia, ARIN supports everybody else, including North and South America, the Caribbean, and sub-Saharan Africa. Currently, there isn't a list of every individual country that falls under ARIN.

Leveraging whois

Armed with this knowledge, you can always find who owns an IP address. This is extremely useful when you are tracking down an IP address that's not resolvable. For example, you find in your logs that an IP address is continually scanning your network for holes. You want to put a stop to this, but how? Often the IP address doesn't have in-addr.arpa entry, so reverse nslookups fail.

With *whois*, you can query any of the three regional registry databases for the IP address' owner. For example, if the IP address is 207.229.165.130. By doing a *whois* on the network block, you can identify the ISP or organization that owns the IP block. Please note that you look up the network block 207.229.165.0, not the specific IP address. Once you find the owner of the IP block, you can then drill down and find the owner of the specific IP. You specify one of the three main registries with -h. The following command asks the ARIN database who owns the network 207.229.165.0:

```
#whois-h whois.arin.net 207.229.165.0
EnterAct, L.L.C. (NETBLK-EACT-BLOCK-1)
3227 N. Sheffield #4R
Chicago, IL 60657
```

```
Netname: EACT-BLOCK-1
Netblock: 207.229.128.0 - 207.229.191.255
Maintainer: EACT
```

Here we learn that the IP address belongs to my ISP, Enteract. This IP block (EACT-BLOCK-1) of 63 class C addresses was received directly from ARIN. If the IP address block belongs to RIPE or APNIC, then the ARIN database will direct you to one of those two. Here is a *whois* lookup of the IP address 195.116.39.59, which is in Poland.

```
#whois-h whois.arin.net 195.116.39.0
European Regional Internet Registry/RIPE NCC
(NETBLK-RIPE-C)
```

These addresses have been further assigned to European users. Their contact information can be found in the RIPE database. Read further to see how to use that database to obtain up-to-date information.

Top level domain names

IP addresses are boring, 32-bit numbers that no one can remember. Domain names, however, are different. These are the highly political entities that countless lawsuits have been fought over. Well, we're going to skip these politics and cover how the technology currently works.

Domain names are how we remember IP addresses. The IP address for our ISP is 206.54.252.8. However, this number is impossible to remember, so we use www.enteract.com, which is much easier to remember and use. But who manages the

domain names, and how does it all work? It all starts with the Top Level Domain name (TLD). Domain names are a hierarchy, with TLDs at the top. Each TLD is then divided into second-level domains, and so on. For example, we'll use the domain name *enteract.com*. COM is the TLD, while *enteract* is the second-level domain name that falls under the TLD COM.

There are two types of TLDs: country-code and generic (gTLD). Every country in the world has a unique two-character identifier, set by ISO 3166 standards. These country-code identifiers are the TLD for each country. For example, US is the TLD for the United States, JP for Japan, and DE for Germany. The following seven generic TLDs also exist: COM, NET, ORG, EDU, MIL, INT, and GOV. Generic TLDs are unique in that they don't denote any nationality.

For every one of these TLDs, both country-codes and generic, there's a specific organization in charge of it, usually called a Network Information Center (NIC). These NICs are responsible for the registration and management of all the second-level domains under the TLD. If you need to find out anything about a second-level domain name, the place to start is the TLD's NIC.

For the country-code TLDs, each country is responsible for its own TLD. Thus, Poland is responsible for its own TLD (PL), just as Japan is responsible for its own TLD (JP). Each country identifies and manages its own NIC, usually via a university or government organization. These country NICs are then authorized by IANA.

The seven generic TLDs are unique in that any organization, regardless of nationality, can use them. The company Network Solutions Inc. is an NIC, thus the name InterNIC, for four gTLDs, COM (commercial), NET (Internet), ORG (organizational—usually not-for-profit), and EDU (educational). The Department of Defense is responsible for MIL (military), the government (actually the Center for Electric Messaging Technologies) for GOV (government), and IANA is responsible for INT (organizations established by international treaties).

To find out who is the NIC for a specific TLD, do a *whois* "TLD"-DOM. The DOM extension tells the whois database to look up a TLD. This will give you the location, point of contact, and the DNS servers of the TLD. Whois, by default, finds this

information at the rs.internic.net database. This database contains the registration information for every TLD. So, to find out who is the NIC for Poland's TLD PL, use the following command:

```
#whois pl-dom
Poland (Republic of) top-level domain (PL-DOM)
Research and Academic Computer Network
Bartycka 18
00-716 Warsaw
POLAND
Domain Name: PL

Administrative Contact:
Krzyszowski, Wiktor (WK856) wiktork@NASK.PL
+48 22 651-05-20..24 (FAX) +48 22 41-00-47
Technical Contact, Zone Contact
Luc, Miroslaw (ML4513) mirek@NASK.PL
+48 22 8268000 (FAX) +48 22 8268009
Domain servers in listed order:
```

BILBO.NASK.ORG.PL	148.81.16.51
COCOS.FUW.EDU.PL	148.81.4.6
SUNIC.SUNET.SE	192.36.125.2
NMS.CYFRONET.KRAKOW.PL	149.156.1.3
DNS2.TPSA.PL	194.204.152.3

Here we see Poland's Research and Academic Computer Network (at www.nask.pl) is in charge of the TLD PL. Also listed are the points of contact, the SOA, and secondary DNS servers. With this information, you can drill down and find information on all second-level domain names under that TLD. After contacting Poland's NIC, we were directed to www.nask.pl/NASK/net/dns-lista.html.

Root servers

Every TLD, both country-code and generic, is also registered with the root server a.root-servers.net. The root server is the absolute top of the TLD hierarchy (represented by a dot "."). It points to the DNS servers of all TLDs. The purpose of a root server is to give the IP address of a TLD's primary or secondary DNS server. When your computer has to resolve a URL, such as www.nask.pl, your computer (if the information has not been cached) will start with the root server. It asks the root server what the DNS servers for the TLD (in this case, PL) are. The root server replies, sending your computer to the TLD's servers, where your system will query about the second-level domain name. Your system then repeats this drill down process until it resolves the URL.

Having a single computer resolving the DNS servers for every TLD isn't a good idea, because of bandwidth and high availability issues. There are 12 other root servers that act as backups to the primary root server. Scattered throughout the world, these 13 servers resolve every TLD. Thus, just like the a.root-servers.net, any of the other 12 root servers act as the ultimate authority for all TLDs. The 13 root servers are listed in **Table A**. (You can get this information by doing a *whois* on the name of the server.)

Registration of second-level domain names

Now that you know how TLDs are managed, what about the second-level domain names—how are those managed? Every TLD is responsible for managing the second-level domain names under it. For example, let's use the most common TLD, COM. This TLD is used the world over, as in *ibm.com* or *toyota.com*. But who controls these second-level domain names, and how are they managed?

If you want to register a second-level domain name with a TLD of COM, you

must do so through Network Solutions Inc.—the company responsible for this TLD (do a *whois* on *com-dom*). Network Solutions Inc. is also responsible for the TLDs ORG, EDU, and NET. To register your second-level domain name, go to their web site at www.internic.net/rs-internic.html. If the second-level domain name you want to use is already registered, then you can't use it. Once the second-level domain name is registered, then the owners are responsible for building and managing their own "NIC" (a primary and secondary server), which resolves the second-level domain name.

The same process is true for any TLD. Say you wanted to register the second-level domain name "this is" with the TLD IT, giving you the web site *www.thisis.it*. First, you'd have to find out who has responsibility of the TLD IT (what country). As we learned earlier, you do this with the following command:

```
#whois it-dom
Italy top-level domain (IT-DOM)
  c/o CNR-Istituto CNUCE
  Via Santa Maria, 36
  Pisa, I-56126 Italy
```

It looks like you'll have to contact the Italian NIC to register your second-level domain name *this-is*. Note, www.ripe.net also provides information on all TLDs in Europe and the Middle East.

Whois for COM, ORG, EDU, and NET

Remember how we did a *whois* on any TLD with the default *whois* database (*rs.internic.net*)? Well, this database also holds information on any second-level domain name under the TLD COM, EDU, ORG, or NET. For example, we'll perform a *whois* on the second-level domain name *intel.com*.

```
#whois intel.com
Intel Corporation
(INTEL-DOM)
2200 Mission College Blvd
P.O. Box 58119
Santa Clara, CA 95052-8119
```

Table A: Root servers

Root	Location
a.root-servers.net	Network Solutions Inc., in Herndon, VA
b.root-servers.net	University of Southern California (ISI), Marina del Rey, CA
c.root-servers.net	Performance Systems International Inc.
d.root-servers.net	University of Maryland, Computer Science Center
e.root-servers.net	NASA Ames Research Center, Moffett Field, CA
f.root-servers.net	Internet Software Consortium, Palo Alto, CA
g.root-servers.net	DOD Network Information Center, Vienna, VA.
h.root-servers.net	Army Research Laboratory, Aberdeen Proving Ground, MD.
i.root-servers.net	Stockholm, Sweden
j.root-servers.net	Network Solutions Inc., Herndon, VA
k.root-servers.net	European Regional Internet Registry, RIPE NCC
l.root-servers.net	University of Southern California (ISI), Marina del Rey, CA
m.root-servers.net	WIDE Project, Fujisawa Japan

Domain Name: INTEL.COM

The reason *whois* will give you this information is that Network Solutions Inc. is responsible for the database rs.internic.net and is the NIC for these gTLDs. Thus, rs.internic.net resolves all TLDs and the second-level domain names for the four gTLDs.

Remember that we can't perform a *whois* on a second-level domain name whose TLD isn't COM, EDU, NET, or ORG. We have to query the TLD's NIC to get information on any second-level domain names. Let's refer to the previous example for the TLD PL. There, we saw that we had to refer to Poland's NIC, nask.pl for information on Poland's second-level domain names.

With the power of *whois*, you can find out who is responsible for any top-level domain name. Once you've identified the NIC of the TLD, you can drill down and find information on second-level domain names under the TLD. Each NIC may have a different method for querying second-level domain names under it. By default, the *whois* server, rs.internic.net, will also answer second-level domain names for the TLDs COM, ORG, NET, and EDU.

Conclusion

There isn't one single organization managing the Internet's resources, specifically IP addresses and domain names. Rather, the Internet is managed in a hierarchical fashion with several organizations at the top. The *whois* command enables you to find out who's managing these resources through the various levels of the hierarchy.

This structure has changed radically over the past several years and will continue to do so. This article captured a snapshot of the Internet at this time. To learn more about the future of the Internet, start with any of the three Regional IP Registries already mentioned or www.gtld-mou.org. ❖

Lance Spitzner is a relative newcomer to the networking world. He learns by blowing up UNIX systems at home. Previously, he was a tank officer in the Army's Rapid Deployment Force, where he blew up things of a different nature. You can reach him at lspitzner@enteract.com. Also, visit Lance's Web site at www.enteract.com/~lspitz.

Password Perls

Automating account creation

by Richard Auletta

Every Solaris system administrator eventually thinks about automating account generation. And, like many system administrators, you have likely started to write such a script or program at one time or another in your favorite programming or scripting language, only to find out you need to generate an encrypted password. This article examines the crux move in an automated account script: the generation of the encrypted password.

Automated account creation

There are many reasons you might want to write a custom automated account creation script. You might want to support account creation from an Internet browser via a CGI script. Or, you might want to set

up a foolproof account generation scheme to be used by support staff. You might even want to create and delete accounts based on a secondary database or with specific defaults and configurations. But surely you'll want to automate the generation of 100 identical accounts that need to be created for a training class one week, and then deleted the next week. In this article we'll show you how to use C, csh, and Perl to write scripts that automate the generation of a set of related accounts.

Existing Tools

You should note that Solaris comes with several tools to help you manage your accounts. The GUI-based `admintool` allows you to create and delete accounts when

sitting at a graphics terminal. You can use the `useradd` and `userdel` applications in terminal mode. Also, you'll find them useful for creating individual accounts, but much less useful when it comes to automating the creation of a large number of accounts. This article will get you well on your way to writing your own custom account creation scripts.

Basics

Creating a new account on almost every UNIX system follows the same basic procedure. The first step is to create the entries in the password and shadow files, including the encrypted password. The second step is to create and initialize the home directory for the account, including setting permissions and copying the needed dot files such as `.profile` and `.cshrc`. As systems have become more complex, these modifications are no longer limited to local directories and changes to a single password file. Today, creating an account might involve creating automount entries and modifying NIS+ tables instead of editing files. But, nonetheless, the idea is the same. You need to create credentials and a home directory for a new user so they can log into their newly created account. In this article, we assume you can manually perform these steps. If you can, then you're ready to automate the steps using a programming language like C, or a scripting language like Perl, or even plain old `csh`.

UNIX passwords

The UNIX password system hasn't changed significantly over the years. The encrypted password is stored in the `/etc/password` file, or on more modern systems such as Solaris, in the `/etc/shadow` file. Of course, on systems running a network information system like NIS+, this data is held in NIS+ tables on an NIS+ server, not in local files. To verify a user, the supplied password is encrypted and compared with the stored encrypted password. The clear text password isn't stored anywhere on the system. When you automate account creation, you'll want the script to generate both a random clear text password to distribute to the user and the encrypted password to be entered into the `/etc/shadow` file or network table.

The other options Solaris offers are to leave the password field empty and allow the user to pick a password at his first login. But this leaves a significant security hole.

`/bin/passwd`

You might be tempted to call the command `/bin/passwd` from your script to set the password, but `/bin/passwd` won't accept a password on the command line. To do so would be a security problem, since the command line arguments are visible to anyone using the `ps` command. You might also be tempted to try passing `/bin/passwd` input data from a script. But, `/bin/passwd` explicitly opens `/dev/tty`, foiling this plan for setting a password for a new account from a simple script. It's possible, however, to get around these issues directly by using `Expect` or `expect.pm` for Perl. But, the fool-proof way is simply writing your own password encryption routine.

Salt and crypt

The C library routine `crypt` is the workhorse that generates encrypted passwords. `Crypt` accepts a salt and the clear text password as its two input parameters—returning a string consisting of the original salt and the encrypted password. The returned string is then placed in the shadow file in the password field of the user's account entry. [Listing A](#) is a very simple C program for calling `crypt`. It takes a password as the only argument, and adds a constant value for the salt, in this case "sa". The program of [Listing A](#) returns the encrypted password on the standard output.

Listing A: C password encryption program

```
#include <stdlib.h>
main(int argc, char *argv[])
{

char *crypt();
void exit();

/* constant salt */
char salt[2] = "sa";

if (argc < 2) exit(EXIT_FAILURE);

/* encrypt and print the password */
(void) printf("%s", crypt(argv[1], salt));

}
```

Once compiled with the command `gcc spass.c -o spass`, the simple program of **Listing A** can be used with a basic csh script to automate account creation. **Listing B** illustrates a very simple script that creates `/etc/passwd` and `/etc/shadow` file entries, creates a home directory, and initializes the dot files and permissions of the user's home directory. It directly adds the entries to the end of the password file, but doesn't check for duplicate entries, doesn't check on input parameters, and doesn't make any record of the account created for distribution to the owner of the account. It also doesn't address working with a network information service such as NIS+. But, it does illustrate the basic steps needed to create an account.

You might also notice that **Listing B's** script displays the problem that `/bin/passwd` works so hard to avoid. By passing the clear text password to a program on the command line, the clear text is made visible to the command `ps`. To avoid this security hole, your script either needs to prompt for the password or generate one randomly. You'll likely prefer having the program generate a random password when automating account

creation. While such scripts can be written in C, sh, or csh, Perl is your better choice due to its greater capabilities and its built-in functions for both the random number and crypt.

Listing B: Account Creation csh script

```
#!/bin/csh
#Arguments: USER_NAME UID GID GCOS PASS_WORD
#Example: smith 100 20 "Dr. Smith" new.pass

#Modify /etc/password
echo $1\:x:$2\: $3\: $4\:/accts/$1\:/bin/csh >> /etc/passwd

#Modify /etc/shadow using C program from listing A
echo $1\:'$pass $5\':::: >> /etc/shadow

#Create home directory
mkdir /accts/$1

#Copy skeleton dot files
cp -r /accts/skel/* /etc/$1

#Change ownership and group
chown -R $1 /accts/$1
chgrp -R $3 /accts/$1
```

Listing C: Random Password Perl Subroutine

```
sub pass_gen {
    local($rnd_passwd, $random_num, @passchar,
    @passspec, $i);

    #
    # Seed the random number generator. XOR the time
    # with the process
    # ID with the checksum of the output of the vmstat
    # -s command.
    # Srand only needs to be called once in the actual
    # script.
    #
    srand (time ^ $$ ^ unpack "%32L*", 'vmstat -s');

    #
    # The password consists of 3 lowercase characters,
    # one special,
    # then 4 lowercase characters
    #
    # Special characters
    @passspec = ('.', '/',
    ',', '%', '#', '@', '!', '>', '<', '+', '=', '0'..'9');
    # Skip l as 1 and l (ell) are very similar and may
    # confuse users.
    @passchar = ('a'..'k', 'm'..'z');

    #
    # Clear rnd_passwd to which characters will be appended to
    # build the password
    $rnd_passwd = "";

    # First 3 characters
    for ($i = 0; $i < 3; $i++) {
        # Generate a random number in the range of @passchar
        $random_num = int(rand($#passchar + 1));
        # Append randomly selected passchar character to
        $rnd_passwd
        $rnd_passwd .= $passchar[$random_num]; }

    # Randomly select a special character from @passspec
    $random_num = int(rand($#passspec + 1));
    $rnd_passwd .= $passspec[$random_num];

    # Final 4 characters
    for ($i = 0; $i < 4; $i++) {
        $random_num = int(rand($#passchar + 1));
        $rnd_passwd .= $passchar[$random_num]; }

    return $rnd_passwd;
}
```

Essential Perl subroutines

Two Perl subroutines are essential when writing an automated account generation script. The first script, shown in [Listing C](#)

Listing D: Encrypted Password Perl Subroutine

```
sub encrypt_passwd {
# Assign passed string to $password
local($password)=@_;

local($random_num, $salt, $i);

#Initialize @salt_chars to hold legal salt characters
local(@salt_chars) = ('a'..'z','A'..'Z','0'..'9','.',',','/');

$salt = "";

for ($i = 0; $i < 2; $i++) {
    $random_num = int(rand($#salt_chars + 1));
    $salt .= $salt_chars[$random_num]; }

return crypt($password,$salt);
}
```

Listing E: Skeleton Perl Account Creation Script

```
#!/usr/local/bin/perl

#
# Check command line arguments.
#
if ($#ARGV != 2) {
    print "Usage: base_name beginning_uid gid\n";
    exit
}

$name = $ARGV[0];
$suid = $ARGV[1];
$gid = $ARGV[2];

# password file data
open(PFILE, ">$name.passwd");

#shadow file data
open(SFILE, ">$name.shadow");

# handouts to user
open(HFILE, ">$name.handout");

# shell script to execute to create directories
open(CFILE, ">$name.cmd");

# Loop to generate 26 accounts
&gen_accts($name,$suid,$gid);
●
```

on page 11, generates random passwords. This particular Perl subroutine returns an 8-character random password consisting of three lower-case characters, one special character, and then four lower-case characters. [Listing C](#) generates a sequence of random characters by initializing the Perl random number generator and then selecting a random character out of the two lists holding legal characters of which the password will consist. The initialization function `srand` is called once with a random value based on the process ID, the time, and the checksum of the output of the `vmstat` command to be sure that unique passwords are generated for each run of the program. Likewise, the Perl script `encrypt_password` of [Listing D](#) generates a two-character random salt and then encrypts the password passed to it that returns the combined salt and encrypted password.

Perl script

Space limitations prevent us from including a complete Perl script for password generation. Besides, this article is about you writing your own custom scripts, not using an existing script. But, we've included some general skeleton code and suggestions to give you ideas for your own script.

At first, you may be tempted to write your scripts to directly modify files and directories (or NIS+ tables), and for some applications this will be necessary. But, you might prefer having your main script create subsidiary scripts that could be run afterwards to perform the actions needed to create and update files. While we're not guaranteeing you won't have problems, subsidiary scripts do allow easier initial debugging of the main script.

The partial script in [Listing E](#) illustrates the basic structure of a complete Perl script. First, it checks for the proper number of command line arguments (in this case, a name for the block of accounts to be created), the starting UID, and the GID. Then, the script opens files to hold the `/etc/passwd` file entries, the `/etc/shadow` file entries, user handouts, and a `csh` script of commands to generate and initialize the home directories. After that, it calls a subroutine `gen_accts` that generates 26 accounts with the form prefix name and an extension a to z.

Listing F is the workhorse script, `gen_accts`. First, it calculates an absolute day 124 days into the future when the accounts should expire for inclusion in the `/etc/shadow` file expiration field. Then, it loops creating 26 accounts of the form `$acct[a-z]` by calculating the UID and writing the user field to the file handle `PFILE` opened in **Listing E**. This file is intended to be manually added to the end of `/etc/password`. Now, it calculates the password and its encrypted form. The encrypted form is written to `SFILE` to be manually appended to the `/etc/shadow` file. Finally, the script calls two additional subroutines that aren't included here. The first subroutine, called `handout`, creates a formatted account form to be given to the user. The second subroutine, called `cshscript`, creates an executable `csh` script that holds the commands for actually creating each account's home directory. This subsidiary script is likely the one that you'll want to customize the most, especially if you're running NIS+ and will need to run `nistbladm` and `nisaddcred` to add password entries and credentials for the new user. Of course, with these examples in hand, you can now write the script of your dreams that directly updates tables and files from the Perl script, and supports a host of site-specific options.

Additional information

1. `/bin/passwd` manual page
 - `man -s 1 passwd`
2. `Crypt` manual page
 - `man -s 3c crypt`
3. `Passwd` file format
 - `man -s 4 passwd`
4. `Shadow` file format
 - `man -s 4 shadow`
5. Perl
 - www.perl.com
6. Expect
 - expect.nist.gov/index.html/

Conclusion

If you're comfortable writing simple shell, Perl, or C programs, you should now be ready to write your own custom account generation scripts. Whether you start by writing simple or fancy scripts, make sure to thoroughly test them. Then, sit back and watch your users' amazement at how fast they can create accounts. ❖

Listing F: Account Generation Perl Subroutine

```
sub gen_accts {
  local($acct,$uid,$gid)=@_;
  local(@ext)='a'..'z';
  local($uidcnt,$i);
  local($cpasswd,$epasswd);
  local($days,$expire);

  $days = int(time/86400);
  $expire = $days+124;

  print "Working on ";

  for ($i = 0; $i < 26; $i++) {
    print "$i ";
    $uidcnt=$uid+$i;
    print PFILE "$acct$ext[$i]:x:$uidcnt:$gid:$acct Group
Account";
    print PFILE ":/accts/$acct/$acct$ext[$i]:/bin/csh\n";

    #Generate cleartext password
    $cpasswd = &pass_gen;

    #Encrypt cleartext password
    $epasswd = &encrypt_passwd($cpasswd);

    print SFILE
"$acct$ext[$i]\:$epasswd\:$days\:\:\:\:$expire\:\n";

    &handout($acct.$ext[$i],$acct,$cpasswd);
    &cshscript($acct.$ext[$i],$acct);

    &flush(STDOUT);
  }
  print "\n";
}
```

Richard Auletta has been creating class accounts for longer than he cares to remember. His current reason for automating class account creation is for the digital and VLSI courses he teaches at the University of Colorado at Denver. You can reach him at rauletta@carbon.cudenver.edu.

Ping the Solaris Dude

by Robert Owen Thomas

Thanks to one and all for the questions! Keep them coming! Submit your Solaris questions to robt@cymru.com.

What the heck is nscd?

The name service cache daemon, or `/usr/sbin/nscd`, is a daemon process that caches name service requests for host lookups, password lookups, and group lookups. It's designed to make lookups faster by caching responses on the local host, thus removing the need for repeat lookups of the same host, user, or group.

For example, a workstation is configured as a DNS resolver. On this workstation, the MUA (mail user agent) uses POP to download E-mail. This could result in a name lookup *every time* the MUA polls for new mail. Not only does this result in increased network traffic, but it also creates an increased load on the DNS name-servers. Multiply this by the number of workstations at a given site, and the result is significant.

With `nscd`, Sun allows you to cache host lookups (as well as user and group lookups) for an amount of time determined in the configuration file. Initial lookups are conducted over the network to remote name service hosts. However, successive lookups are conducted locally. It's fast and efficient!

How does nscd work?

`Nscd` works through the `libc` interfaces, such as `gethostbyname()` and `getpwnam()`. `Nscd` checks calls to these `libc` interfaces are for cached values. For example, if a query for a hostname finds a cached value, the cached value is returned. If no entry is found, the query follows the path determined by the `/etc/nsswitch.conf` file.

`Nscd` uses the doors IPC mechanism—a new and fast IPC methodology introduced by Sun. `Syslogd` also uses doors. The file `/etc/.name_service_door` is the door's IPC file for `nscd`, so don't delete it!

Also, `nscd` keeps an eye on the local configuration files, `/etc/passwd`, `/etc/hosts`, and `/etc/group`. For example, if a change is made to `/etc/hosts` (and this is one of the host databases as defined in `/etc/nsswitch.conf`), `nscd` will invalidate all host entries within its cache in approximately 10 to 20 seconds. Note that `nscd` keeps a distinct cache for each of its monitored databases, so the invalidation of one cache will not affect the other two. There's one catch: `nscd` doesn't watch `/etc/nsswitch.conf`. If you change `/etc/nsswitch.conf`, you'll need to stop and restart `nscd`.

Sun has also added some strong security features to `nscd`. `Nscd` doesn't intercept `getspnam()` calls, so passwords aren't actually cached in the user space of `nscd`. Also, the `nscd` startup script checks the permissions of the host, `passwd`, and `group` tables if NIS+ is used. If those tables aren't open to unauthenticated users, caching is disabled.

Administering nscd

There are several key command line options you can pass to `nscd`. Fortunately, `nscd` is smart enough to avoid launching multiple instances of itself. So, go ahead and enter these commands as often as you like—you won't end up with lots of `nscd` daemons running on your system.

If you have made a change to the host's database, for example, and you want `nscd` to immediately invalidate the host cache, simply enter:

```
orc# nscd -i hosts
```

This will clear the host cache. You could, of course, substitute `passwd` or `group` for host above.

If you decide you don't want groups cached, for example, simply enter:

```
orc# nscd -e group,no
```

Note: That's a comma between `group` and `no`. Make sure that you don't place a space between them.

Your group cache is now disabled. To re-enable, simply change the no to a yes.

The most important option, -g, is also the only option available to non-root users. This is the statistics reporting option and a must if you're going to tune nscd. As an example, we've cut out just the password cache statistics here:

```
orc# nscd -g
nscd configuration:

    0 server debug level
"/dev/null" is server log file

passwd cache:

Yes cache is enabled
153 cache hits on positive
↳entries
    0 cache hits on negative
↳entries
10 cache misses on positive
↳entries
    0 cache misses on negative
↳entries
93% cache hit rate
    0 queries deferred
    6 total entries
211 suggested size
600 seconds time to live for
↳positive entries
    5 seconds time to live for
↳negative entries
    20 most active entries to be
↳kept valid
Yes check /etc/
{passwd,group,hosts} file for changes
No use possibly stale data
↳rather than waiting for refresh
```

The statistics look good. We have a 93 percent hit rate for our passwd cache, with only 10 misses. Note that many of the configuration parameters previously discussed are reported with nscd -g.

Configuring nscd

Now for the tricky part—configuring nscd. Note that the default configuration of nscd is likely robust enough for most sites and systems. Further using the nscd options (-i, -e) will likely meet any of your unique needs. For those of you that require some fine tuning (or just want to experiment), we look at /etc/nscd.conf, the configuration file

for nscd. Let's take the logfile, debug-level, and passwd cache entries as an example:

```
# Section of /etc/nscd.conf:
# logfile /dev/tty
debug-level 0

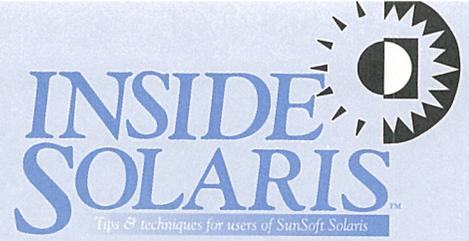
positive-time-to-live
passwd 600
negative-time-to-live
passwd 5
suggested-size
passwd 211
keep-hot-count
passwd 20
old-data-ok
passwd no
check-files
passwd yes
```

The top entry (which is commented out) is the logfile entry. You can place the name of a logfile here, if you wish to send the debug output to a file. A nice trick to remember, though, is /dev/tty. This sends all the debug output to stdout, which can be very handy during debugging and testing sessions.

The next entry is the debug level, in a range from 0 to 10. The output doesn't become interesting until debug level 10, but we'll talk about that later.

The positive-time-to-live entry is the number of seconds that a successful match (e.g., finding user joe in the passwd file) will remain in the cache. Increasing this number will raise your cache hit rate, but might play havoc with the integrity of the cache. If the remote NIS master should have a change, and the local positive-time-to-live entry is set at twelve hours, the systems will quickly become out of sync. If your cache hit rate is around 90 percent or greater, we'd suggest leaving this alone.

The negative-time-to-live entry is the number of seconds that an unsuccessful match (e.g., can't find user joe in the passwd file) will remain in the cache. This is



Inside Solaris (ISSN 1081-3314) is published monthly by ZD Journals.

Customer Relations

US toll free (800) 223-8720
Outside of the US (716) 240-7301
Customer Relations fax (716) 214-2386

For subscriptions, fulfillment questions, and requests for group subscriptions, address your letters to

ZD Journals Customer Relations
500 Canal View Boulevard
Rochester, NY 14623

Or contact Customer Relations via Internet E-mail at zdjcr@zd.com.

Editorial

Editor Garrett Suhm
Copy Editors Rachel Krayter
Christy Flanders
Taryn Chase
Contributing Editor Rainer Dorsch
Lance Spitzner
Richard Auletta
Robert Owen Thomas
Print Designers Lance Teitsworth, Kristi Weese

General Manager Jerry Weissberg
Editor-in-Chief Joan Hill
Editorial Director Michael Stephens
Managing Editor Kent Michels
Circulation Manager Renee Costanza
Print Design Manager Charles V. Buechel
VP of Operations and Fulfillment Michael Springer

You may address tips, special requests, and other correspondence to

The Editor, *Inside Solaris*
500 Canal View Boulevard
Rochester, NY 14623

Editorial Department fax (716) 214-2387

Or contact us via Internet E-mail at sun@zsjournals.com.

Sorry, but due to the volume of mail we receive, we can't always promise a reply, although we do read every letter.

Postmaster

Periodicals postage paid in Louisville, KY.

Postmaster: Send address changes to

Inside Solaris
P.O. Box 35160
Louisville, KY 40232

Copyright

© 1998, ZD Journals, a division of Ziff-Davis. ZD Journals is a trademark of Ziff-Davis. *Inside Solaris* is an independently produced publication of ZD Journals. All rights reserved. Reproduction in whole or in part in any form or medium without express written permission of Ziff-Davis is prohibited. ZD Journals reserves the right, with respect to submissions, to revise, republish, and authorize its readers to use the tips submitted for personal and commercial use.

Microsoft, Windows, Windows NT, and MS-DOS are registered trademarks of Microsoft Corporation. All other product names or services identified throughout this journal are trademarks or registered trademarks of their respective companies.

Price

Domestic \$115/yr (\$11.50 each)
Outside US \$135/yr (\$16.95 each)

Back Issues

To order back issues, call Customer Relations at (800) 223-8720. Back issues cost \$11.50 each, \$16.95 outside the US. You can pay with MasterCard, VISA, Discover, or American Express.

ZD Journals publishes a full range of journals designed to help you work more efficiently with your software. To subscribe to one or more of these journals, call Customer Relations at (800) 223-8720.

To see a list of our products, visit our Web site at www.zsjournals.com.

SunSoft Technical Support

(800) 786-7638

Please include account number from label with any correspondence.

at least as important as the positive-time-to-live, because it prevents the system from constantly searching for bad accounts. Again, tuning this value can have detrimental effects that could result in your systems becoming out of sync.

The suggested-size value is the number of hash buckets for the given cache. This value should be tuned only if nscd -g reveals that the number of entries in the cache has grown to four times the size of the suggested-size value.

The keep-hot-count entry is the number of entries nscd will keep up-to-date. Consider this your top 10 (or 20) list. This value can be tuned to the number of approximate entries you expect to see per day.

The check-files variable enables or disables the file-checking feature of nscd. If enabled, nscd will poll the file associated with the cache (/etc/passwd, /etc/group, or /etc/hosts) approximately every ten seconds. Unfortunately, this poll interval can't be tuned. The only significant performance gain you're likely to see in disabling this feature is if the database files are accessed over an NFS mount. We don't recommend disabling this feature.

Debugging with nscd

Here is a sample of the debug output using debug level 10 and a logfile of /dev/tty (stdout). This debug session is capturing a remote login of user robt:

You can see all of the steps necessary to log in a user: checking the tty, changing tty ownership, checking mail, opening files such as .profile, etc. This also demonstrates the benefit of nscd. Without this caching mechanism, and if this system were an NIS client, each of these checks would result in a request going over the network to a remote NIS server. That's a lot of overhead just to log into a system.

Not only can the nscd debugging output help track down issues with nscd, it can also be used as a debugging tool for other issues

```

Wed May 28 23:43:58.2986 nscd debugging
enabled
Wed May 28 23:43:58.2986 Start of new
logfile /dev/tty
Wed May 28 23:44:17.0975 getpw_lookup:
looking for name root
Wed May 28 23:44:17.1042 getpw_lookup:
nscd FOUND passwd name root
Wed May 28 23:44:17.1355 getgr_lookup:
looking for name tty
Wed May 28 23:44:17.1380 getgr_lookup:
nscd FOUND group name tty
Wed May 28 23:44:17.1522 gethost_lookup:
looking for address 192.168.0.2
Wed May 28 23:44:17.1629 gethost_lookup:
nscd FOUND addr 192.168.0.2
Wed May 28 23:44:19.3069 getpw_lookup:
looking for name robt
Wed May 28 23:44:19.3088 getpw_lookup:
nscd FOUND passwd name robt
Wed May 28 23:44:21.9377 getpw_lookup:
looking for name robt
Wed May 28 23:44:21.9390 getpw_lookup:
found name robt in cache
Wed May 28 23:44:21.9412 getpw_lookup:
looking for name robt
Wed May 28 23:44:21.9420 getpw_lookup:
found name robt in cache
Wed May 28 23:44:21.9454 getgr_lookup:
looking for name tty
Wed May 28 23:44:21.9461 getgr_lookup:
found name tty in cache
Wed May 28 23:44:21.9957 getpw_lookup:
looking for uid 201
Wed May 28 23:44:21.9970 getpw_lookup:
nscd FOUND uid 201
Wed May 28 23:44:22.0396 getgr_lookup:
looking for name mail
Wed May 28 23:44:22.0418 getgr_lookup:
nscd FOUND group name mail
Wed May 28 23:44:22.0444 getpw_lookup:
looking for uid 201
Wed May 28 23:44:22.0452 getpw_lookup:
found uid 201 in cache
    
```

as well. Remember that there are three caches: hosts, passwd, and group. Others may be added in the future.

In conclusion, nscd is a powerful addition to Solaris. With some basic understanding of the features and tuning tips, it's easy to derive great benefit from nscd. ❖

Rob Thomas is an aspiring blues guitarist earning his living as a UNIX and networking consultant. You can reach him at: robt@cymru.com. Also, visit Rob's Web site at www.cymru.com/~robt.



Printed in the USA.
This journal is printed on recyclable paper.