

Inside Solaris™

Tips & Techniques for users of Sun Solaris

MP3 audio

by Richard Auletta

M P3 is revolutionizing how music and audio are stored, played, sold, and exchanged. MP3 is a music compression scheme that will compress audio by a factor of 10 while still maintaining CD quality audio, turning a 40 MB CD audio track into a manageable 4 MB MP3 file. This article will get you started creating and playing MP3s on your Sun Sparc Solaris or x86 workstation.

Why MP3?

Everyone has a different reason for creating MP3s. I like the convenience of creating custom tapes for my car and leaving my audio CDs at home when listening to music on my SparcStation at work. Of course, eight hours of music on a single recordable CD-ROM is reason enough for any music fan to embrace MP3. And, for those of you on the move, portable and automotive MP3 players are available for downloading the MP3 files you create.

What's MP3?

MP3 is technically the MPEG Layer-3 perceptual audio coding scheme. The coding scheme compresses the audio in a manner that exploits the properties of the human ear and can

achieve high levels of compression while still maintaining audio quality.

Coding and compression are needed because audio sampled at the 16-bit 44.1-kHz rate of CDs uses 1.4 MB to store one second of audio data. At that rate, even a 9 GB hard disk can only hold about 14 full-length CDs.

In comparison, MP3 uses only 16 KB per second of audio for CD quality sound. If sound quality isn't the primary objective, MP3 can produce FM radio quality decoded audio at a compression factor of 24, and telephone quality can reach compression factors of nearly 100. Because MP3 uses perceptual coding techniques that consider how we hear, the decompressed audio will sound better than audio that was sampled at a lower sample frequency and resolution. In other words, MP3 encoded 16-bit audio will result in better-sounding audio than comparable uncompressed 8-bit audio.

Preliminaries

The first thing you need to be aware of is that strict copyright laws cover most recorded music. You can only create MP3s for your personal use from most commercial sources. With that said, there are three steps to creating MP3 audio from an audio CD as illustrated in **Figure A**: ripping, encoding, and playback.

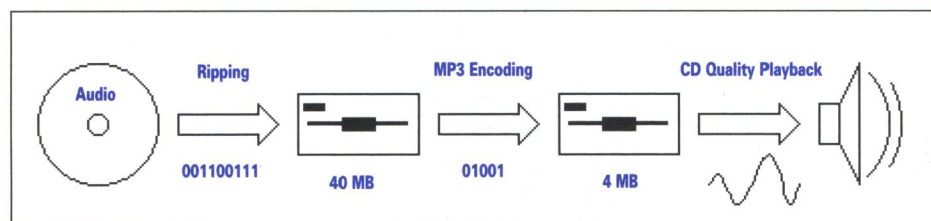


Figure A: Creating and playing MP3s involves these three steps.

In this issue:

- 1** MP3 audio
- 3** Y2K and BIND configuration
- 4** File system full
- 8** Using Solstice DiskSuite 4.2
- 12** A controlled date-change facility for Y2K environment

16 Quick Tip: **The 10sec. TIME SAVER** Yesterday

Ripping is the process of copying audio data from an audio CD to your hard disk. While it's possible to use an analog input to capture the audio, the direct digital transfer avoids the introduction of any noise into the audio stream from the conversion between the analog and digital domains.

Encoding is the process of converting the raw audio data, typically a WAV or AIFF file, into an MP3 file. Encoding is computationally intensive and takes about six minutes on a 300-MHz UltraSparc for a four-minute 44.1-kHz 16-bit stereo audio track ripped from an audio CD.

Playback, of course, is where the MP3 is decoded and the audio recovered. The decoding process is considerably simpler than the encoding, and can be performed in real-time. The focus of MP3 players tends to be on user features, such as managing play lists and their look and feel.

The section on Web Resources will point you to where you can obtain the programs described in the rest of this article. All the programs described in this article come as binaries, so you can just download and enjoy.

Ripping

Galette is a classic UNIX command line utility that will rip audio tracks off your worksta-

tion's CD-ROM player. Usage couldn't be easier. Insert the audio CD into your CD-ROM player and type `galette -t 3 -f AIFF track3.aiff`. This usage will rip track three from the CD and store it as an AIFF file named `track3.aiff`. To rip the entire CD in one stroke, use the form `galette -a -f AIFF tunes`, and *Galette* will consecutively number each track ripped. Some audio rippers will query a CDDb database to obtain the album title and track information.

Galette has proven to be very reliable. The only problem we've had is with enhanced audio CDs that contain a Windows FAT file system. *Galette* won't rip the last track on these disks. This appears to be a problem with `cdio ioctl` interface on Solaris, as Volume Manager is also known to fail on some enhanced audio CDs. There's a GUI interface available for *Galette* on the Cybersoft Web page.

Encoding


The Solaris port of the DOS-like Windows *BladeEncoder* MP3 encoder works as advertised under Solaris, readily converting audio files to MP3. Usage can be as simple as `bladeenc track3.aiff`, which will produce a new MP3 encoded file called `track3.mp3`. GUI interfaces are available for *BladeEncoder* on its homepage for those who prefer a point-and-click interface.

Of course, you can encode any audio source to MP3. *BladeEncoder* accepts WAV and AIFF input formats. *Sox*, the Sound eXchange program, will convert between many formats, including Sun AU to WAV and AIFF formats, allowing you to encode a range of audio sources.

Playback

Xaudio and its motif-based graphical interface `mxaudio` allows you to play back and enjoy the MP3s you have created on your Solaris Sparc or x86 workstation. *Xaudio* also has an SDK for those who would like to write their own interface to the *Xaudio* decoder or include MP3 playback directly in a custom application.

Enjoying

If you've been playing audio CDs with `xmcd` or `workman`, you might want to consider the convenience of encoding them as MP3s for your personal use, and in the process freeing up both your CD-ROM drive and your briefcase for things other than audio CDs. 

Web resources

- The canonical MP3 site www.mp3.com/
- Galette CD ripper www.cybersoft.org/
- BladeEnc MP3 encoder home8.swipnet.se/~w-82625/
- Xaudio MP3 decoder and SDK www.xaudio.com/
- Sox Sound eXchange www.spies.com/Sox/
- Audio CD database www.cddb.com/

Coming up...

- Sun freeware
- Quick file backups

Y2K and BIND configuration

by Paul A. Watters

The Berkeley Internet Name Domain (BIND) is the standard Internet name resolution server and client software used on Solaris systems. Since Solaris is widely used as the operating system for running primary domain name services (DNS) on the Internet, any potential issue regarding Year-2000 compliance and DNS should be taken seriously. In this article, we'll look at the configuration of BIND, and how conventional use of the `serial` parameter in BIND for registering DNS revisions will cause problems after the year 2000. This is *not* a Y2K bug in the BIND software, but a problem that arises from using `serial` parameters in a way that's most useful to us.

BIND background

Making Internet configuration accessible was the primary motivation behind DNS in the early years of the Internet. Although our Solaris systems are quite happy processing network information based on numeric IP addresses, trying to remember many IP addresses is cognitively taxing. In addition, matching domain names to IP addresses makes it easy to locate services on the Internet (for example, when we want to check out the latest information from Sun, we can easily guess the host name for the Sun Web server is www.sun.com).

The primary DNS server in any domain uses four separate files to store information about hosts in that domain. The cache initialization file (`named.ca`) has a list of root servers from which authoritative DNS information can be retrieved. The localhost file (`named.local`) associates the primary DNS server with its own loopback address (127.0.0.1), and is the cornerstone for retrieving hostnames on the local subnet. The reverse domain file (`named.rev`) translates IP addresses into hostnames by using pointer (PTR) records, whereas the domain hosts file (`named.local`) contains address (A) information for all hosts in the domain.

The latter three files are collectively known as *zone* files, and are kept on the primary DNS server for each Internet domain. Among the other parameters that make up the zone files, such as address and pointer records, the Start of Authority (SOA) records contain several

parameters that are used primarily by DNS servers outside the domain to retrieve information about hosts within the domain (that is, secondary servers provided by your ISP). These include `refresh`, which is the time in seconds that a secondary server should wait before updating zone files from the primary server, and `expire`, which is the time in seconds that a secondary server should retain zone information about the domain before discarding it.

The problem

The parameter we're interested in is `serial`, which is a serial number that *must* increase in value every time local zone information is updated (that is, when adding a host to the domain, or changing an IP address). In BIND, this is an eight-digit number that can either start from zero and increase to 99999999 (for example, more than ninety-nine million DNS changes in the zone), or more commonly, as a four-field, two-digit number that's composed of year, month, day, and daily revision number. For example, if the SOA for `named.local` had a serial of 99020701, this would identify the file as the first DNS revision on February 7, 1999.

Anyone who has followed the debate over Y2K problems can immediately see the dilemma: following the convention outlined above, the serial for `named.local` on January 1, 2000 would be 00010101. Since this number is much less than the serial as at December 31, 1999 (99123101), no zone information will be updated by secondary servers. Thus, unless an alternative way of creating serials is introduced before December 31, 1999, no DNS updates will be registered in many domains around the world.


Solutions

There are three possible solutions to the problem. The first solution is that the BIND software could be changed to use a ten-digit number for serials, such as 2000010101 for January 1, 2000. One of the problems with such a change would be the large number of sites that would have to recompile and restart their primary name servers at some specified date so that zone changes would be recognized.

This should be a long-term goal for those involved in BIND development, and would retain the useful date / revision format for identifying zone revisions.

The second solution is that, on January 1, 2000, primary DNS administrators can continue to add serials in a linear fashion from 99123101, so that further revisions take the numbers 99123102, 99123103 and so on. This provides for at least 876,897 DNS revisions, hopefully by which time a new standard for DNS serials has been agreed upon and an implementation delivered.

The third solution is to start using linear serial addition *today*, rather than leaving it until December 31st. The reason is that the number of possible DNS revisions increases dramatically for every month that we can save: using the example for February 7th mentioned previously, the number of possible changes is 979,298, an increase of around 12 percent on the December 31st figure.

In any case, administrators will need to act before December 31, 1999, if they intend to ever change their primary DNS information in the future. 



File system full

by Jeff Forsythe, Sr.

Your file systems may become overburdened in one of several ways. They become fragmented, the real estate isn't used correctly, or they become full. This article will deal mainly with this last issue, a full disk. Most of what we'll discuss is just plain old common sense. But until you've been face to face with this problem, you might not give the issue the priority it demands. Be forewarned however, you *can* begin this procedure now and avoid the *File System Full* message. I suggest you do so now, proactively, before you have to do it out of necessity. So let's look at what can be done.

Unnecessary files

Removal of unnecessary files, such as temporary, junk, and test files, is usually a good start. Production data files don't require duplicates, so these can safely be removed. "What?" you say, "Remove production files?" Yes, remove all copies of production files. If they're *really* production files, then there should only be one, not 10 copies.

Sometimes copies may be needed in order to make changes, but these are no longer production files, they're work files. Make sure you know the difference, and don't remove work files. Work files should be in file systems and directories specially designed for work files and not in production locations. And if you use the *vc.sh* that was introduced in

"Shell toolbox 101, part 2" in our January, 1999 issue, you won't have to worry about mistaking these for production files, because older copies have the version number appended to them. Just make sure that you really want to remove these older files. Remember, when you configured *vc.sh*, if you chose to keep five previous versions of each file, there was probably a reason.

Semi-duplicates

So, how do you go about identifying the candidates for removal? The first such files are semi-duplicate files. These are files with duplicate or semi-duplicate names that contain nearly, but not-quite, duplicate contents. Semi-duplicates are easy to find, albeit time consuming. You simply run the *find* command to a temporary file, run the *sort* command with the *unique* flag against the temporary file and output that to a second temporary file, and finally run the *compare* command against both temporary files. The resulting output is a list of potential semi-duplicate filenames. This code snippet finds those files that are named the same within the same file system:

```
find /fileys > tmp1
sort -u tmp1 > tmp2
comm -23 tmp1 tmp2 >
    semi-duplicates
```



Do this in / and you've got all duplicate named files on your system. You must look at the potential files manually to see if they really are semi-duplicates and not just duplicate names for totally unlike files. If you have a better way, please contact me!

But what about the files that you copied to other directories or have in the same directory but have made a minor name change to (such as .old, .bak, etc.)? As administrators you wouldn't be at fault, because we know you're using the vc.sh and would never have this problem. But your users do this all the time, causing you this full disk problem. With a slight modification to the find command, you can expand and fine-tune your search for semi-duplicates. You can do so especially through judicious use of the -name flag and a star, such as:

```
find /filesystem -name "base*" >
  tmp1
```

which finds all filenames that begin with *base* in the file system.

Duplicates

The next candidates for removal are the real duplicates—actual duplicate copies of files spread across the same directory, different directories, or even across file systems. This will seemingly take forever to run. But, if you really want to clean house—and you should periodically—then this approach will guarantee to clean out duplicate files.

To find duplicates, run `get_dups.sh`. It's extremely slow since it's performing a find and a checksum on all files in the given path, but it will find all of them for you. Keep in mind that if you have a lot of files to check, this may require a lot of disk space in the `tmpdir`. If you're running it to help clean out a full or almost full file system, make sure to put your temporary files elsewhere so that it can run. You should also have a lot of system-time free when you want to do this. We suggest you don't have this running when your backup is running. In fact, it's always a good practice to have nothing running during a backup.

Different processors will run this at various speeds. Some will work better if you run one process for each file system and then concatenate the resulting files together before sorting out the duplicates.

The output from this tool is a text file listing all duplicated files and their relative path to

the location you searched. Don't forget that this list contains *all* the duplicates, meaning it contains the original file you'll want to keep, too. So don't try to automate the removal process unless you don't mind performing restores from your last good backup. A nice enhancement would be to add the full path in by replacing the dot in the find command, or at least by adding a heading saying what the file is a listing of. We'll leave that to you.

Found them, now what?

Of course, once the semi-duplicates and duplicates are found, you have to use your own judgement on whether or not they can be removed. Local political climate will dictate who has the final say on this matter, unless you have ultimate authority over your system—in which case you can just delete them.

Don't forget to clean up after yourself. Those temporary files you just built are space wasters, too. The `get_dups.sh`, shown in [Listing A](#) on page 6, will clean up its working files, but leaves the resulting list. Once you've cleaned these up, remove the list file. The semi-duplicates temporary files will also be hanging around. You can use a special directory especially for your temporary files, perhaps `/tmp/admin-tmp`. If so, make sure that it gets cleaned out automatically in order to save you time and effort. However, it's always a good policy to have this directory cleaned out after a backup. A good time to do this is at the end of your backup script, or to append it to the backup command in cron. The best method would be to verify that the backup was successful prior to cleaning up these files. If the backup failed, don't remove anything. You might need the files to figure out what happened or to rebuild something.

Other culprits

Other culprits in wasting disk space are large files. You can find them using `du` and `sort`, or by using the find command with the `-size` flag. You must work with your users to determine what's removable. Perhaps you'll get lucky and find some really big files that can be removed. Some users will intentionally waste space by creating large junk files.

Another thing to remember is that users will get files in weird places, such as `/usr` or `/etc`. A good plan for these semi-static file systems is to have a list of the original file systems' contents on hand and to periodically

Listing A: The `get_dups.sh` script will remove duplicate files

```
#!/bin/sh
#####
#
# SHELL:      get_dups.sh
# DATE WRITTEN: 03/13/1999  JAF, Sr.
# DATE UPDATED:
# PURPOSE:    Find duplicate files
# USAGE:      get_dups.sh [filesystem] [-t tmpdir]
# FLAGS:      -t : assign temporary directory
# ARGUMENTS:  filesystem : File system or directory
#             in which to search for duplicate files
#             -t tmpdir : Assigns tmpdir as the
#             place to put the temporary work files
# RETURNS:    0 - Nominal
#             1,2,3 - Usage errors
# CALLS:      N/A
# CALLED-BY:  N/A
# ERRATA:     None
# LIMITATIONS: Inherently SLOW due to find and sum of
#             all files.
#####

# Process arguments
case $# in
  0) ;;
  1) DIR=$1;;
  2) if [ "$1" = "-t" ]
      then
          TMPDIR=$2
      else
          echo "USAGE: get_dups.sh [dir|filesystem] [-t tmpdir]"
          exit 1
      fi;;
  3) DIR=$1
      if [ "$2" = "-t" ]
          then
              TMPDIR=$3
          else
              echo "USAGE: get_dups.sh [dir|filesystem] [-t tmpdir]"
              exit 2
          fi;;
  *) echo "USAGE: get_dups.sh [dir|filesystem] [-t tmpdir]"
      exit 3;;
esac

# Set defaults
HERE=`pwd`
DIR=${DIR:=${HERE}}
TMPDIR=${TMPDIR:=${HERE}}

# Change to DIR
cd ${DIR}

# Build the file list w/check sums
find . -exec sum -r {} \; > ${TMPDIR}/chksum

# Get sorted list of check sums only
awk -F"." '{print $1}' ${TMPDIR}/chksum | sort > ${TMPDIR}/
↳/srtsum

# Get unique list of sorted check sums
sort -u ${TMPDIR}/srtsum > ${TMPDIR}/uniqsrtsum

# Get unique list of duplicate scheck ums
comm -23 ${TMPDIR}/srtsum ${TMPDIR}/
↳uniqsrtsum | sort -u > ${TMPDIR}/uniqdups

# Set the Field Separator to <CR>
IFS=""
"

# Create header for duplicates list
echo "get_dups.sh `date +%m/%d/%Y` `date +%H:%M`" >
↳${TMPDIR}/dups
echo " " >> ${TMPDIR}/dups

# Create list of duplicates
for LINE in `cat ${TMPDIR}/uniqdups`
do
    grep "${LINE}" ${TMPDIR}/chksum >> ${TMPDIR}/dups
    echo " " >> ${TMPDIR}/dups
done

#
# BEEN THERE - DONE THAT - NOTE:
#
# Remember that the dups list contains ALL the duplicates
# including the one you will want to keep so don't automate
# the removal beyond this point without considering that.
#

# Clean up temporary files
rm -f ${TMPDIR}/chksum ${TMPDIR}/srtsum ${TMPDIR}/
↳uniqsrtsum ${TMPDIR}/uniqdups

# Change back to original directory
cd ${HERE}

exit 0
```

check for lost sheep. You should use a shell script to check for changes to these file systems. A report will tell you any changes that were made (using the `sum` command), any additions to the file system, and any deletions from it. This is an especially important script, and one that deserves its own mention in an upcoming article.

This isn't a procedure you'll want to use for dynamic file systems where your users traditionally keep data. Be careful when removing files from system directories and file systems, because the file you remove could be the one that's keeping your system up and going. Don't forget whose playground you're on when you're visiting `/`, `/usr`, and the like—you're on root's turf. And if you mess up, root wins, you lose, and the users suffer.

Your users' perceptions may take up quite a bit of disk space. They sometimes think they need copies of files for extra protection. Typically, they have multiple copies on the same file system with the same and/or different names and spread across multiple file systems with the same name. Semi-duplicates and duplicates have already been discussed. Finding multiple copies with different names requires a good rapport with the users. Discerning their personal naming conventions can be tough. If open communication and a trusting relationship can't be established with the users, then you have bigger problems than a full file system.

Disk management policies


In addition to the preceding ideas, you must ask yourself and your users if there's a better way to manage the amount of disk space at your disposal. Can you use links instead of copies? Can you compress your data (which may give astonishing results)? Is there a different format the users can use that requires less space? For instance, CAD users may be saving their files in ASCII format instead of binary, costing much more disk usage.

Using the existing structure to the best of your advantage is also important. If you have free space on another file system, then you might be able to utilize it by moving and using data there or moving the data there and linking to it, which is invisible to the users. Hopefully, these ideas will help to prolong your free disk space long enough to get that new disk in place.

Ounce of prevention

From an administrator's viewpoint, once you have a new disk in place, there are a few things you'll want to consider. Built-in waste (by you) is an administrator's friend. Don't compress until you must; use a space-wasting format when you can (save CAD in ASCII on purpose), etc. These are the tools you'll need to help get you through the next disk crisis. Above all, don't advertise the fact that you're wasting, or it will come back to haunt you.

In addition to the unallocated disk space, you should already have set aside space for normal growth. Actually creating a hidden file system and not mounting it can be a good idea. You might also want to put a large junk file in it that uses a great deal of its resources, to make it less appealing to someone else who happens upon it. You can create a nice junk text file (one full of numbers and special characters is good, because they aren't questioned as often as alphabetic text files). You can just create a little loop sending numeric strings to a file until it gets huge. When the space becomes necessary, you just remove the junk file, a portion of the contents, or delete the entire file system and re-allocate the space where it's required. You must decide how much to set aside, if any.

Hopefully you don't need any of these ideas, but when you do, they'll prolong your dwindling resources for a little while. If that doesn't help, beg your boss to let you buy an extra disk drive to sit on a shelf for just such emergencies. In any case, start cleaning up your disks now—don't wait until you have no choice! 

Inside Solaris back issues

Back issues of *Inside Solaris* are a handy resource. If your *Inside Solaris* library is incomplete, you should invest in back issues. These are available for \$9.00 each (\$11.00 each outside of the US). To order back issues, you can contact our Customer Relations Department by sending an email to zdjcr@zd.com. You can also contact us by phone at (800) 223-8720.

Using Solstice DiskSuite 4.2

by Arthur Haigh

Avoiding a system crash is always a top priority for system administrators. You can significantly increase the availability of your system by creating a mirrored root partition using Solstice DiskSuite. *Mirroring* is a Redundant Array of Independent Disks (RAID) technique that forces data to be written to two identical disks or partitions, creating continuously up-to-date, online backups of important partitions like root.

Mirroring a disk or partition isn't difficult to implement or understand. Sun gives you the tools you need to configure and understand RAID in the Solstice DiskSuite package. If you have the Solaris System 7 server distribution, then you have a copy of everything you need. DiskSuite 4.2 is included on the Solaris Easy Access Server 2.0 Software CD-ROM.

Deciding on a RAID configuration

There are basically five levels of RAID (plus a few hybrid variations). Mirroring, which is discussed in this article, is known as RAID level 0. When choosing a RAID configuration, one must balance the variables (that is, fast, cheap, or reliable) and optimize the configuration at the expense of one of the other choices. For example, mirroring provides high reliability (because you have two copies of everything), but is expensive (because you need twice the number of disks).

You can learn much more about RAID by installing and reading the AnswerBook packages from the Solaris Easy Access Server 2.0 Documentation CD-ROM. You can also access Sun's documentation Web page using a Web browser. Go to <http://docs.sun.com/> and click the link for Systems Administration. Then, search for "disksuite." This will take you to the Solaris DiskSuite 4.2 Collection. Don't choose DiskSuite 4.0 by mistake.

Laying the groundwork

Conceptually, RAID isn't difficult. There is, however, some jargon that you'll need to become familiar with. Here are a few definitions that Sun uses for describing RAID:

- **Metadevice**—A virtual disk or device that represents the collection of a group of physical slices or disks.
- **State database**—A database stored on disk that keeps track of the state of the metadevices, error conditions, and configuration information.
- **Mirror**—A metadevice made up of subsets of other metadevices (sub-mirrors). Data are duplicated on each submirror.
- **One-way mirror**—A mirror with only one submirror.

Getting started

To begin, I'm going to make some assumptions about this installation. Please note that these are simplifying assumptions, and not requirements. If your needs differ, don't let these assumptions pose limitations:

- I'm starting with two identical disks with identical partition maps, as shown in **Table A**. You, of course, can choose to use different maps, but make appropriate modifications to these instructions where needed. The small partitions in **Table A** are set aside for the metadevice state database and replicas.
- This is an initial installation of DiskSuite 4.2 (that is, there are no older versions on the host).
- Your system has a local CD-ROM drive.
- Both disks are internal and are on one controller. We'll use `/dev/dsk/c0t0d0`, and `/dev/dsk/c0t1d0` in our examples.

Installing DiskSuite 4.2

Locate the Solaris Easy Access Server 2.0 Software CD-ROM, and mount it. If the volume manager daemon (`vold`) isn't running, then mount the CD-ROM manually.

```
# mount -F hsfs -o ro /dev/dsk/c0t6d0s0  
➔ /cdrom/solaris_easy_access_srvr_2_0
```

If the volume manager daemon is running, then the CD-ROM will mount automatically. Now, change directories.

```
# cd /cdrom/solaris_easy_access_srvr_2_0
```


Use the package add utility to install DiskSuite as follows:

```
# cd sparc
# pkgadd -d `pwd`
```

The following packages are available:

```
1 SUNWmd      Solstice DiskSuite
  (sparc) 4.2,REV=1998.02.09.12.47.28
2 SUNWmdg     Solstice DiskSuite Tool
  (sparc) 4.2,REV=1998.14.09.08.19.32
3 SUNWmdn     Solstice DiskSuite Log Daemon
  (sparc) 4.2,REV=1998.02.09.12.47.28
```

Select package(s) you wish to process (or
 ➤ 'all' to process
 all packages). (default: all) [?,??,q]:

Type *all*, and then answer the questions about running the installation scripts as root. If you want to use the graphical user interface instead of the pkgadd utility, then you can run the WebStart Installer by executing the following command:

```
# ./installer
```

After the packages are installed successfully, reboot the host:

```
# /usr/sbin/reboot
```

When the system is up, you can verify your installation by invoking the package information utility:

```
# pkginfo | grep SUNWmd
system      SUNWmd      Solstice
➤DiskSuite
system      SUNWmdg     Solstice
```

```
➤DiskSuite Tool
system      SUNWmdn     Solstice
➤DiskSuite Log Daemon
```

For convenience, set the search path to include the newly installed DiskSuite executables and man pages. For the Bourne shell (/bin/sh), use these commands:

```
# PATH=/usr/opt/SUNWmd/sbin:$PATH
# MANPATH=/usr/opt/SUNWmd/man:$MANPATH
# export PATH MANPATH
```

For the C shell (/bin/csh), enter these commands:

```
# set path = (/usr/opt/SUNWmd/sbin $path)
# setenv MANPATH /usr/opt/SUNWmd/man:$MANPATH
```

If you choose to set the paths, then you don't need to type the full path to the commands, as I've done in the examples that follow.

Creating the state databases

When configuring two RAID disks, Sun recommends creating two databases per disk. You can see in [Table A](#) that I've created a small partition, slice 3, on each of the disks. This slice isn't for a filesystem, but for the state database and its replicas.

Create the initial state database and replicas using the metadb command:

```
#!/usr/opt/SUNWmd/sbin/
➤metadb -a -f -c 2 c0t0d0s3 c0t1d0s3
```

where *-a* specifies that the initial state database should be created. The *-f* forces the creation and the *-c 2* creates two copies on each disk.

Table A: Sample partition table configured for root, swap and replica database partitions.

Part	Tag	Flag	Cylinders	Size	Blocks
0	root	wm	0-2642	1.92 GB	(2643/0/0) 4017360
1	swap	wu	2643-2729	64.57 MB	(87/0/0) 32240
2	backup	wm	0-2732	1.98 GB	(2733/0/0) 4154160
3	unassigned	wm	2730-2732	2.23 MB	(3/0/0) 4560
4	unassigned	wm	0	0	(0/0/0) 0
5	unassigned	wm	0	0	(0/0/0) 0
6	unassigned	wm	0	0	(0/0/0) 0
4	unassigned	wm	0	0	(0/0/0) 0

You can use the `metadb` command to verify that the databases were created:

```
# /usr/opt/SUNWmd/sbin/metadb

flags first blk      blockcount
au   16      1034    /dev/dsk/c0t0d0s3
au  1050    1034    /dev/dsk/c0t0d0s3
au   16      1034    /dev/dsk/c0t1d1s3
au  1050    1034    /dev/dsk/c0t1d1s3
```

Specifying `metadb -i` will print this information and include descriptions of the flags.

Creating the metadevices

Next, we'll need to create the mirror metadvice. To do this, we'll first create the submirrors, one for each disk. Then we'll initialize the mirror and attach one submirror to it, creating a one-way mirror. We'll reboot the system from the one-way mirror metadvice and then attach the second submirror, allowing the mirrors to synchronize.

Create the submirrors

The root partition is on `/dev/dsk/c0t0d0s0`. Create the first sub-mirror, which we'll call `d1`:

```
#/usr/opt/SUNWmd/sbin/
↳metainit -f d1 1 1 c0t0d0s0
d1: Concat/Stripe is setup
```

The `-f` option forces the initialization on a mounted file system; `d1` is the name of this submirror. This submirror has one stripe and is one stripe wide (as indicated by the `1 1`). Finally, we specify the device `c0t0d0s0`.

Now, create the second submirror called `d2`:

```
#/usr/opt/SUNWmd/sbin/
↳metainit d2 1 1 c0t1d0s0
d2: Concat/Stripe is setup
```

Note that this command looks similar to the first `metainit` command. The differences are that the first time we issued it, we specified `-f`. That isn't needed here. Also, note that we specify `d2` instead of `d1`. Finally, note the device is different. Here we specify the second internal disk (target 1).

Initializing the mirror

Now that we've created the submirrors, we can initialize the mirror and then attach the

first submirror, `d1`. Since, for now, there will be only one submirror in the mirror, we call it a one-way mirror. First, initialize and attach submirror `d1` to the new mirror `d0`:

```
#/usr/opt/SUNWmd/sbin/metainit d0 -m d1
d0: Mirror is setup
```

Now we have a metadvice, `d0`, which is a mirror containing one submirror, `d1`.

We'll need to reboot the host from the new metadvice, `d0`, instead of from the device `/dev/dsk/c0t0d0s0`. The `metaroot` command changes the `/etc/vfstab` and `/etc/system` files to enable us to do this. First make copies of `/etc/vfstab` and `/etc/system`:

```
#cp /etc/vfstab /etc/vfstab.orig
#cp /etc/system /etc/system.orig
#/usr/opt/SUNWmd/sbin/metaroot d0
```

Sun recommends running the `lockfs` command prior to rebooting:

```
#/usr/sbin/lockfs -fa
#/usr/sbin/reboot
```

During the boot process you may see the following messages:

```
WARNING: forceload of misc/md_trans failed
WARNING: forceload of misc/md_raid failed
WARNING: forceload of misc/md_hotspares failed
```

The Sun DiskSuite Installation and Product Notes documents (in the General Information section) state that, "these warnings are harmless, and may be ignored."

After the system has successfully rebooted from metadvice `d0`, take a look at the `/etc/vfstab` file to see the changes that `metaroot` produced. You'll notice that `/dev/dsk/c0t0d0s0` has been replaced with `/dev/md/dsk/d0`. Also, look at the mounted root file-system:

```
#df -k /
Filesystem      kbytes    used    avail
↳capacity Mounted on
/dev/md/dsk/d0  1946246  635686  1252173
↳34% /
```

Notice that we're now referencing the metadvice and no longer referencing the physical device. Our next task is to attach the second submirror, `d2`, to `d0`:

```
#!/usr/opt/SUNWmd/sbin/metattach d0 d2
d0: submirror d2 is attached
```

You may be able to hear the disks chattering, indicating that the submirrors are synching.

That's it. You have just mirrored your root partition. Now the mirror metadvice can be treated as a normal device. It can be mounted, unmounted, and you can run `fsck` or `ufsdump`.

Creating a mirror for swap, /usr and /opt

The creation of mirrors for other unmountable filesystems is straightforward. Had you used a different partition table from the one given in [Table A](#), perhaps specifying `/usr` and `/opt` as separate partitions, you could use these steps to mirror those slices. In this example, we'll mirror swap.

Creating the swap submirrors

The swap partition is on `/dev/dsk/c0t0d0s1`. Create the first submirror:

```
#!/usr/opt/SUNWmd/sbin/
↳metainit -f d11 1 1 c0t0d0s01
d11: Concat/Stripe is setup
```

Now, create the second submirror:

```
#!/usr/opt/SUNWmd/sbin/
↳metainit d12 1 1 c0t1d0s01
d12: Concat/Stripe is setup
```

Initialize the mirror and then attach the first submirror, `d11`:

```
#!/usr/opt/SUNWmd/sbin/metainit d10 -m d11
d10: Mirror is setup
```

Now, we have a swap metadvice, `d10`, which is a mirror containing one submirror, `d11`.

We'll need to reboot the host and use the new metadvice `d10` for swap instead of `/dev/dsk/c0t0d0s1`. We used the `metaroot` command when we mirrored root. For swap, as well as `/usr` and `/opt`, we need to edit the `/etc/vfstab` file by hand. First make a copy of `/etc/vfstab`:

```
#cp /etc/vfstab /etc/vfstab.noswap
```

Edit `vfstab`, substituting `/dev/md/dsk/d10` in place of `/dev/dsk/c0t0d0s1`. Then reboot:

Note: In the event that the metadvice or the primary submirror becomes corrupted, you may need to revert to booting from the second internal disk (`c0t1d0s0`). This is known as the alternate boot device. Determine the alternate boot device by looking at the links in the `/dev` directory:

```
# ls -alF /dev/rdisk/c0t0d0s0
lrwxrwxrwx  1 root  root
↳67 Feb 10 17:00 /dev/rdisk/c0t1d0s0 ->
↳ ../.. /devices/sbus@1f,0/espdma@e,
↳8400000/esp@e,8800000/sd@0,0:a,raw
```

Notice that the path starts with `../..`—which means the path is really

```
/devices/sbus@1f,0/espdma@e,8400000/
↳esp@e,8800000/sd@1,0:a,raw
```


Take out a piece of paper and your logbook and write this down. Tape it to the host. Hopefully, you'll never have to think about it again. If your boot disk fails, you'll need to halt the system and boot from the alternate device. Refer to the *DiskSuite AnswerBook* documentation to learn more. You should learn what to do if a metadvice fails before it actually fails.

```
#!/usr/sbin/reboot
```

Now attach the second submirror, `d12`, to `d10`:

```
#!/usr/opt/SUNWmd/sbin/metattach d10 d12
d10: submirror d12 is attached
```

That's all there is to it. You've mirrored the most difficult partitions—the unmountable filesystems. It will be simple for you to mirror your data or file partitions, since these are mountable systems. Now you can go back to the documentation, review what we've done, and, if you want to mirror mountable systems, read about that, as well. You should be aware that there's a graphical user interface that's designed to simplify the use of *DiskSuite*. It comes with the installation, and it's called *DiskTool*. Personally, I find the command-line technique easier to use.

Through the application of these few simple commands, you've significantly increased the reliability of your system. Backups are still vital, however. You can further increase the availability of the host by configuring the submirrors on separate SCSI controllers, removing the single point of failure on controller `c0`. 

A controlled date-change facility for Y2K environment

by Atiq Hashmi

With the year 2000 approaching, many companies are working hard to make their products Y2K compliant. A necessary part of this Y2K work is providing the testing environment where the products can be tested with the system clock rolled forward into the year 2000 and beyond. Generally, in a software release cycle, the developers and testers simultaneously share the resources on the development and test environments, respectively.

However, exclusive access of the system date is one important need for a Y2K release testing, and this may require frequent involvement of system administrators, as many users may need such exclusive access. Some approaches involve scheduled updates of Y2K dates or dividing the machine usage among users for exclusive access. Simple approaches include making the standard date program, `setuid root`, or giving users root privilege, etc.

Where the access need is intermittent and not clearly defined, these approaches are often inconvenient and unsafe for one or more of these reasons:

- They don't provide controlled access, so users can step on each other and reset the date, affecting another user's work.
- They allow anybody to access the system date, whether or not they're supposed to, thus potentially affecting any test in progress.
- They require elaborate planning and scheduling, which may not always fit well with the daily changing work activities of the users.

In this article, we'll provide a scheme to offer a date-change facility that will allow multiple users to share the system date resource by having exclusive access to it one at a time—avoiding the aforementioned problems, as well as providing a totally user-managed environment. The idea revolves around the concepts of the UNIX `setuid root` feature, locking mechanism, and an authenticated list of users.

The `setuid root` feature

Every file on a UNIX system has a set of permission bits. These include access bits for the *user*, *group*, and *other* user types and additional bits for special permissions, including the sticky bit and `setuid` bit.

Normally, when a process runs an executable program, it runs it (if the program has appropriate execute permissions) under the effective user ID of the owner of the process, even though the program may be owned by another user. Turning the `setuid` feature on allows the process to execute the program under the user ID and privilege of the owner of the program. This is useful, for example, in the `passwd` program, which needs to modify the root owned `/etc/passwd` file.

Setting the `passwd` program as `setuid` allows a user process running the `passwd` program (to change his/her password) to temporarily assume the root's effective user ID and modify the `/etc/passwd` file. Our idea is to install a copy of the `date` program in an appropriate directory as `setuid root`, instead of setting the system file (`/usr/bin/date`) as `setuid`. The following command sets a program `setuid` (the 4 makes it `setuid`):

```
#chmod 4755 <program>
```

Locking system date

The facility provides a locking mechanism by using a lockfile that can only be removed by the file owner or root. The need is to allow multiple users to create a lock file in a directory, but only allow the owner to delete it. This is achieved by the sticky bit, which, when set on a directory, allows removal of the file only if the user has write permission on the directory and either owns the file, owns the directory, or is the super-user. This control is obtained with

```
#chmod 777 <lockfile dir>
#chmod u+t <lockfile dir>
```

The first line makes it writable by everybody, so that all users can create a lock file, thus locking the date-change capability. The

DOWNLOAD

ftp.zdjournal.com/sun

second line sets the sticky bit on the directory to prevent other users from removing the lockfile. The lockfile itself must be created with mode 744 for this to work. To change the system date, the script takes an argument (which must be in the format accepted by the date program) for the new date, which is simply passed on to the date program.

Authenticated user list

To control the list of users who can change the date, the script uses a file containing the user ID of each user, one per line. If a user isn't on the list, access isn't given. The ownership of this file can be given to one user (for example, the lead tester) who could manage the list, avoiding the need for administrator intervention.

Listing A shows a shell script that implements our controlled date change facility. There are some global variables at the top that can be changed as appropriate. The main part of the script handles the options to lock/unlock the date, modify the system date, and check who holds the lock and help option. The usage message shows how to specify the new date value.

Setting up your system


The setup requires root privilege. It's better to install this facility in a commonly accessible

place like /usr/local. **Listing B**, on page 15, shows an example setup. Note that some of these commands may not be necessary as they are done from a root account, but are mentioned for understanding.

The script also allows a user to put a note in the lock file for other users, for example, how long he would need the lock for etc. When a user invokes the date.y2k.sh script, it asks if he wants to write a message. If he does, it opens the lockfile for writing the message; otherwise, it simply creates a blank file. This file serves as the lock on the date-changing capability until the user or the root user unlocks it using the -u option.

Conclusion

The ideas discussed here offer a convenient way to allow non-root users to handle the system date resource on a Y2K environment in a controlled and safe manner. At the same time, it saves the user and system administration time to handle frequent requests for date or user list updates.

The tool can also be modified to handle client/server type applications spanning multiple machines by installing it in a common mounted filesystem and making appropriate changes to the script. 

Listing A: Our controlled date change script

```
#!/usr/bin/ksh
# This script allows non-root users to
# change the system date on a Y2K machine.

USAGE="\n
syntax: date.y2k.sh [-hluca] [-m <date>] \n\
where \n\
\tformat is: mmddHHMM[yy]y.SS \n\
\t-h : help \n\
\t-l : lock the system date \n\
\t-u : unlock the system date \n\
\t-c : check who has the date lock \n\
\t-m : modify the system date \n\
e.g. \n\
to set date to Oct 23 12:55:42 2024,\n\
use: \n\
$ date.y2k.sh -u 1023165524.42 \n\
Note, hours is in GMT.\n\
Please unlock date when done.\n"

DATEDIR=/usr/local/y2kdatedir/bin
LOCKDIR=/usr/local/y2kdatedir/lockdir
WHOAMI=`/usr/ucb/whoami`
LOCKUSERS=/usr/local/y2kdatedir/users.allowed
CURLOCKID=""
RM=/bin/rm

function islockmsg
{
    echo "\nUser '$CURLOCKID' has lock."
    echo "Contact that user to unlock it."
    if [ -s $CURLOCKID.lock ]
    then
        echo "\nHere is a note from $CURLOCKID:\n"
        cat "$LOCKDIR/$CURLOCKID.lock"
    fi
}

function datelock
{
    arg="$1"
    case "$arg" in
```

Listing A: *continued*

```
-c) if [ -f $LOCKDIR/*.lock ]
then
    islockmsg $CURLOCKID
    exit
fi
;;

-u) if [ -f $LOCKDIR/*.lock -a "$WHOAMI"!="$CURLOCKID" ]
then
    islockmsg $CURLOCKID
    exit
fi

$RM $LOCKDIR/*.lock
if [ $? -eq 0 ]
then
    echo "\n lock by '$WHOAMI' turned *OFF*.\n"
fi
;;

-l) if [ -f $LOCKDIR/*.lock -a "$WHOAMI" = "$CURLOCKID" ]
then
    echo "\nYou already hold lock with the message(if
any):\n"
    cat $LOCKDIR/*.lock
    exit
elif [ -f $LOCKDIR/*.lock ]
then
    islockmsg $CURLOCKID
    exit
fi

echo "Want to write a note for other users(y/n) [n]?:"
read x
if [ "$x" = "y" -o "$x" = "Y" ]
then
    vi $LOCKDIR/$WHOAMI.lock
else
    touch $LOCKDIR/$WHOAMI.lock
fi
chmod 744 $LOCKDIR/$WHOAMI.lock
if [ -f $LOCKDIR/$WHOAMI.lock ]
then
    echo "\n lock turned *ON* exclusively for '$WHOAMI'\n"
else
    echo "\n locking for $WHOAMI failed\n"
fi
;;

-m) if [ -f $LOCKDIR/*.lock -a "$WHOAMI"!="$CURLOCKID" ]
then
    islockmsg $CURLOCKID
    exit
fi
touch $LOCKDIR/$WHOAMI.lock
;;
esac
}

#-----main-----
if [ $# -eq 0 ]
then
    clear; echo $USAGE
    exit
fi

grep $WHOAMI $LOCKUSERS > /dev/null
if [ $? -ne 0 ]
then
    echo "\nYou are not allowed, please contact
administrator.\n"
    exit
fi

cd $LOCKDIR
CURLOCKID=`echo *.lock | cut -d"." -f1` #used
globally

while getopts lmuch: opt
do
    case $opt in
        l) datelock -l
            ;;
        m) datelock -m
            $DATEDIR/date.y2k -u "$2"
            ;;
        c) datelock -c
            ;;
        u) datelock -u
            exit
            ;;
        h | *)
            clear
            echo $USAGE
            exit
            ;;
    esac
done
```

Listing B: Sample actions to create our date-change environment

```
#cd /usr/local
#mkdir y2kdatedir                #install location
#chmod 755 y2kdatedir            #make it read-only by others
#cd y2kdatedir
#mkdir bin lockdir                #create two directories
#chmod 755 bin
#chmod 777 lockdir
#chmod u+t lockdir               #set sticky bit on lockdir
#cp /usr/bin/date bin/date.y2k   #make a copy of date program
#chown root bin/date.y2k         #make root the owner
#chmod 4755 bin/date.y2k         #make date.y2k setuid
#vi users.allowed                #enter userids one per line
#chmod 755 date.y2k.sh           #make it read and execute
```

Get ZDTips!

If you haven't already done so, be sure to sign up for ZDTips, a FREE service that delivers software tips right to your email account. In addition to tips for Sun Solaris, ZD Journals offers ZDTips on Internet development, application development, and desktop applications. Just visit our site at www.zdtips.com and make your selections.

About our contributors

Richard Auletta is currently helping to construct the silicon chips that pave the information superhighway. He can be reached at rauletta@orci.com.

Jeff Forsythe, Sr. started programming in 1978. He's worked with Solaris and other UNIX flavors in retail, banking, manufacturing, and currently with the federal government. He welcomes your comments, code fixes, administrative scripts, etc. You can reach Jeff via email at forsythe@tuscan.net.

Arthur Haigh is the senior systems administrator in the Division of Nuclear Medicine, School of Medicine, University of Pennsylvania. He can be reached at art@rad.upenn.edu.

Atiq Hashmi is a software engineer at Telcordia Technologies (formerly Bellcore). His interests are in system and network administration, applications and tools as well as internet related technologies. He can be reached at hash100@mail.eclipse.net.

Paul A. Watters is a research officer in the Department of Computing, Macquarie University, Australia. He can be reached at pwatters@mpce.mq.edu.au.

Inside Solaris™

Tips & Techniques for users of Sun Solaris

Inside Solaris (ISSN 1081-3314) is published monthly by ZD Journals 500 Canal View Boulevard, Rochester, NY 14624.

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 email at zdjcr@zd.com.

Editorial

Editor Garrett Suhm
Assistant Editor Jill Suhm
Copy Editors Rachel Krayner
Christy Flanders
Taryn Chase

Contributing Editors Richard Auletta
Jeff Forsythe, Sr.
Arthur Haigh
Atiq Hashmi
Paul A. Watters
Print Designer, Cover and Content Design Lance Teitsworth

VP of Content Development Ed Passarella
VP & Publisher, Content Development Linda Edwards
Executive Editor Kent Michels
Circulation Manager Jeff Marcus
Manager of Design & Production Stacy Dobles
Manager of Design Services Charles V. Buechel
Executive VP of Operations Jerry Weissberg
Director of Customer Support Douglas Neff
Director of Operations & Fulfillment Kress Riley

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 email 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 92880
Rochester, NY 14692

Copyright

Copyright © 1999, ZD Inc. ZD Journals and the ZD Journals logo are trademarks of ZD Inc. *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 ZD Inc. 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.

Inside Solaris is a trademark of ZD Inc. Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, SunOS, SunInstall, OpenBoot, OpenWindows, DeskSet, ONC, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. Other brand and product names are trademarks or registered trademarks of their respective companies.

Printed in the USA.

Price

Domestic \$99/yr (\$9.00 each)
Outside US \$119/yr (\$11.00 each)

Our Canadian GST# is: R140496720. CPM# is: 1446703.

Back Issues

To order back issues, call Customer Relations at (800) 223-8720. Back issues cost \$9.00 each, \$11.00 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.

PERIODICALS MAIL

Sun Technical Support
(800) 786-7638

C:7661905 00002096 04/00
CLINTON TOWNSHIP, MI 48035-4218

Please include account number from label with any correspondence.

QUICK TIP Yesterday


by Jeff Forsythe, Sr.

DOWNLOAD

ftp.zdjournal.com/sun

The **10** sec.
TIME SAVER

Quick, what was yesterday's date? OK, that was good, if you gave the right answer. But how quickly can your computer give

yesterday's date? In the blink of an eye with the code in **Listing A**. You can find this script on our ftp site as yesterday.sh. 

Listing A: Code for getting yesterday's date

```
return_and_exit()
{
    echo "${YESTERDAY_MONTH}/${YESTERDAY_DAY}/${YESTERDAY_YEAR}"
    exit 0
}

TODAY_MONTH=`date +%m`
TODAY_DAY=`date +%d`
TODAY_YEAR=`date +%Y`

if [ "${TODAY_DAY}" != "1" ]
then
    YESTERDAY_MONTH=${TODAY_MONTH}
    YESTERDAY_DAY=`expr ${TODAY_DAY} - 1`
    YESTERDAY_YEAR=${TODAY_YEAR}
    return_and_exit
fi

case ${TODAY_MONTH} in
    1) YESTERDAY_MONTH=12
        YESTERDAY_DAY=31
        YESTERDAY_YEAR=`expr ${TODAY_YEAR} - 1`
        return_and_exit;;
    2) YESTERDAY_MONTH=1
        YESTERDAY_DAY=31
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    3) YESTERDAY_MONTH=2
        if [ `expr ${TODAY_YEAR} - 1992 % 4` = 0 ]
        then
            YESTERDAY_DAY=29
        else
            YESTERDAY_DAY=28
        fi
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    4) YESTERDAY_MONTH=3
        YESTERDAY_DAY=31
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    5) YESTERDAY_MONTH=4
        YESTERDAY_DAY=30
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    6) YESTERDAY_MONTH=5
        YESTERDAY_DAY=31
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    7) YESTERDAY_MONTH=6
        YESTERDAY_DAY=30
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    8) YESTERDAY_MONTH=7
        YESTERDAY_DAY=31
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    9) YESTERDAY_MONTH=8
        YESTERDAY_DAY=31
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    10) YESTERDAY_MONTH=9
        YESTERDAY_DAY=30
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    11) YESTERDAY_MONTH=10
        YESTERDAY_DAY=31
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
    12) YESTERDAY_MONTH=11
        YESTERDAY_DAY=30
        YESTERDAY_YEAR=${TODAY_YEAR}
        return_and_exit;;
esac
```