ZD JOURNALS

December 1999 Volume 5 Number 12
www.zdjournals.com/sun US $9.00

Inside Solaris™

Tips & Techniques for users of Sun Solaris

Cross-platform development made easy with MetaCard 2.2

by Clayton E. Crooks II

Those of us who are given the dreaded cross-platform development assignment can now breathe a little easier. We have a new weapon at our disposal. With the release of MetaCard 2.2, cross-platform development has become an easy, all-inclusive, and affordable reality.

At first glance, MetaCard appears as many other application development tools. As you can see in **Figures A** and **B**, it appears much like Visual Basic on the Windows platform, as it contains a GUI IDE, scripting language, script editor, and script debugger. It includes built-in support for databases such as Sybase, Access, and Oracle, has multimedia abilities comparable to products like IMSI Multimedia Fusion or Macromedia Director, and hypermedia support like a dedicated Web development environment.

When we first began looking over the package, we were concerned that maybe too many things were placed into one product. We've all seen instances where a product has been given too many features instead of focusing on its main function. This isn't the case with MetaCard. Although the IDE is very vanilla-looking, it flawlessly integrates all of the tools into one seamless development environment. **Figures C** and **D**, on page 2, demonstrate the context sensitive documentation and examples that we felt were particularly pleasing.

The MetaTalk language, which is compatible with the HyperTalk and SuperTalk languages used with HyperCard and SuperCard, respectively, is the easiest, multipurpose scripting language that we've ever used. Along with the basic functions, additional extensions built into MetaTalk include HTTP support, arrays, and regular expression pattern matching.

MetaCard also supports a full range of features required for User Interface development. **Figure E**, also on page 2, shows a simple example of some of these controls, which include the basics such as buttons, comboboxes, or tabs. There's also built-in support for pop-up menus, dialog boxes, and floating palettes.

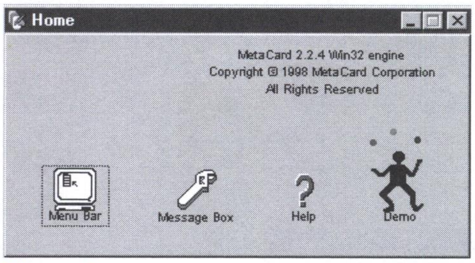

Home
MetaCard 2.2.4 Win32 engine
Copyright © 1998 MetaCard Corporation
All Rights Reserved
Menu Bar
Message Box
Help
Demo

Figure A: *Home is the basic element of the MetaCard IDE.*

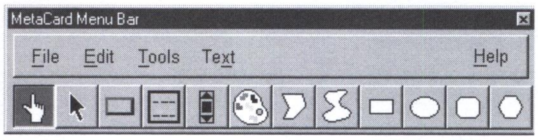
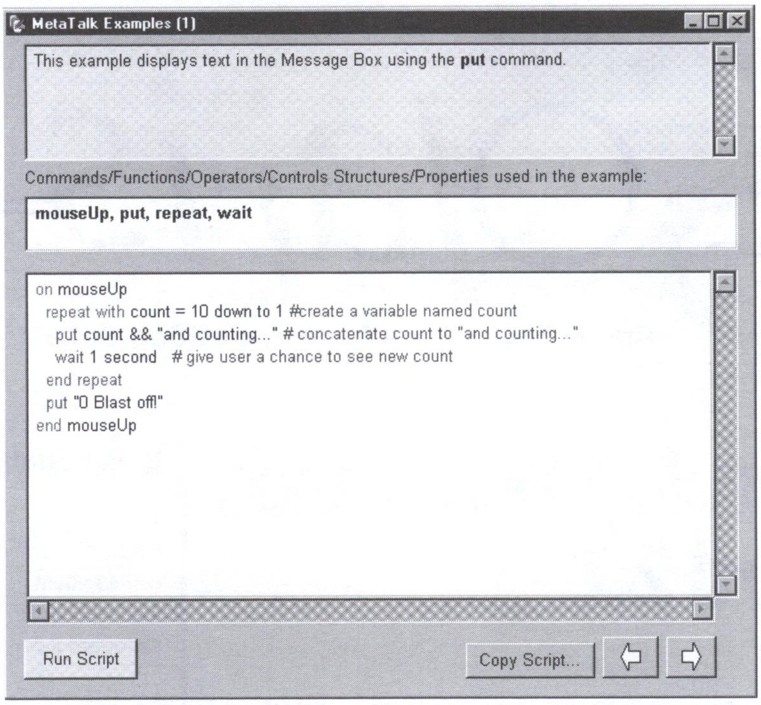


Figure B: *An additional part of the MetaCard IDE is the MetaCard Menu Bar.*

In this issue:



This example displays text in the Message Box using the **put** command.

Commands/Functions/Operators/Controls Structures/Properties used in the example:

mouseUp, put, repeat, wait

```
on mouseUp
  repeat with count = 10 down to 1 #create a variable named count
    put count && "and counting..." # concatenate count to "and counting..."
    wait 1 second   # give user a chance to see new count
  end repeat
  put "0 Blast off!"
end mouseUp
```

Figure C: *Here's a MetaTalk example script from the context Help system.*

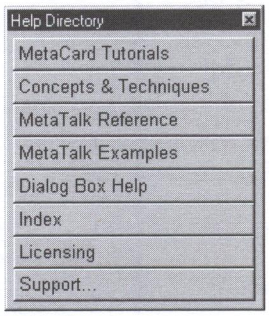


Figure D: *The Help Directory is part of MetaCard's excellent Help system.*

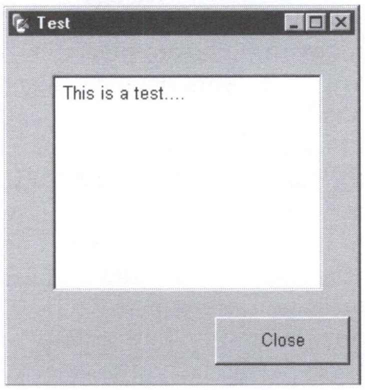


Figure E: *Buttons and text boxes are just some of the built-in MetaCard components.*

The language itself is much easier to learn than conventional programming languages such as Visual Basic or Delphi, but is also powerful. In fact, the development environment has been written entirely in MetaCard itself.

MetaCard applications are distributed with MetaCard engines that are comparable in performance to Java virtual-machine-based applications. All OS-specific routines are contained in these separate modules enabling the easy porting of applications. Because the product has been developed with cross-platform abilities in mind, it either lacks support completely or has very basic support for COM/OLE/ActiveX.

Although this product is more than sufficient for development, MetaCard isn't complacent about the future. Future releases are planned with Internet-related improvements like email support for POP3 and SMTP and FTP file transfer. The already powerful MetaTalk language will also be receiving some upgrades. The extended commands will provide a more full-featured object-oriented programming environment, which will allow development of larger-scale applications. OODBMS features, such as multi-user access, version control, and distributed data management, are currently being designed as well.

Conclusion

For those of us who are given the task to develop something for many platforms, MetaCard (**www.metacard.com**) is definitely a product that's up to the challenge. With a $995 price tag for a single user, any-platform environment, you'll definitely want to consider this package for your development needs.

Coming up...

- The /proc file system
- The /etc/system file
- Developing Solaris knowledge bases
- Priority paging

advice from the EXPERTS

Solaris and Y2K: better hurry!

by Clayton E. Crooks II

Calculations based on a two-digit format can possibly yield incorrect results once the value rolls over to 2000, or 00. It would introduce a new level of ambiguity in what is being represented by these two digits.

Testing for this problem can be very exhausting and frustrating. According to Sun Microsystems, Solaris 7 and up are Y2K compliant out of the box. Older versions 2.4, 2.5, 2.5.1, and 2.6 can be made Y2K safe by applying a set of Y2K patches. In **Table A**, you'll find a list of sites containing these patches and other Web sites of interest. These patches should be installed prior to any testing. If you're uncertain as to which version of Solaris you have, at the console or terminal prompt type *uname -a* to determine the SunOS version, or refer to your original CD-ROM.

Testing

Once you've acquired and applied the OS patches for your respective system, it's time to begin the test phase of your task. Be careful with Year 2000 testing. Adhering to the following Sun-created guidelines will eliminate many possible problems.

- If the date command is used to set the time into the future, reboot the machine before starting any tests. Use a separate test system.
- Running tests before the machine has been rebooted with a new date can give confusing results. Be careful of cron jobs and any processes running during the date change.
- Don't enable password aging on the test system, or you'll run into trouble when you set the date forward.
- Don't use your own workstation. Unexpected things can happen to files and email while testing.
- When testing is finished, be sure you delete all files you have touched or, better yet, reformat disks and reinstall the OS.
- Don't mount home directories or other writeable NFS directories except those absolutely needed for tests.

Table A: *Web sites of interest*

Sun's Y2K home page	**www.sun.com/y2000/index.html**
Sun OS patches	**http://sunsolve.sun.com**
The unofficial Solaris guide	**www.solarisguide.com**

- Be sure to delete touched files in NFS-mounted directories.
- Don't forget to check filenames that start with a period (by using the `ls -al` command).

As Sun suggests, it's best to completely restore the box, including a reload of the OS, reformatting disks, and then reload any touched apps, etc. You run the risk of corrupting data or the box by not performing this task. This may affect file time stamps, system logging, and backups, and cause other undesirable effects.

Rollover, reboot, and day-of-week tests

The first step in testing should be the rollover, reboot, and day-of-week tests. The Rollover test will check the December 31, 1999 to January 1, 2000 rollover with the power on. Set the date to 31-dec-1999, and set the time to 23:59. Observe the system date after 00:00 A.M. The day-of-week testing will use the previous testing information to check the day of the week. Verify that the date is 1-Jan-2000 from the previous test, and observe the system day-of-week display.

The date retention test will check for date retention on the machine after rollover and rebooting. Verify that the date is 1-Jan-2000 from the previous test. Power down the system, and then power up the system. Observe the system date.

The next step will be to check date rollover from December 31, 1999 to January 1, 2000 with the power off. Set the date to 31-Dec-1999, and set the time to 23:50. Power down the system before it can roll over to year 2000. Wait until after midnight with the power off. Power up the system. Observe the system date.

Testing the manual date setting

The next series of tests deal with testing of the manual date setting. Manually set the system date to 1 Jan 2000, and observe the system date.

Next, check to see if the date settings are retained. With the date still in the year 2000, power down the system, and then power up the system. Observe the system date.

For the leap year date setting, set the date to 29-Feb-2000. Observe the system date. To test the leap year rollover with the power on, set the date to Monday, 28-Feb-2000. Set the time to 23:59, and observe the system date after midnight.

To test the leap year retention setting with the power off, verify that the date is 29-Feb-2000 from the previous test. Power down the system, and then power up the system. Observe the system date.

To test the leap year rollover for February 29—power on, verify the date is 29-Feb-2000 from the previous test. Set the time to 23:59, and observe the system date after 00:00 A.M. Set the date back to the correct time. We recommend restoring the entire system from tape.

Conclusion

With some basic hardware and OS testing out of the way, you've cleared a major hurdle in your path to Y2K readiness. You should follow up with all software/hardware vendors to discuss their Y2K issues, as you may find it necessary to test these individually as well.

In Table A, you'll find a list of Sun's Year 2000 alert services. You will definitely want to sign up for this to stay up to date with Sun's Year 2000 news and announcements. All registrants will automatically receive Sun's monthly email newsletter *Sun Customer Millennium (Y2K) Chronicle*, as well as notifications on other major Sun Year 2000 announcements and updates.

QUICK TIP

Yesterday version 2.0

by Jeff Forsythe, Sr.

Have you ever needed yesterday's date in a script? In the July 1999 issue of *Inside Solaris* the Quick Tip "Yesterday," presented a script to return yesterday's date. A reader wrote to us asking why the presentation was so rigid and bulky. The reader also suggested putting the date in a file and using it from there.

In response, we're going to revise the script so that everyone can understand it. Our solutions aren't always the most elegant, but they're designed to get the point across.

Some of our mainframe counterparts think there should only be one way of doing things. The *beauty* of Solaris (and all UNIX-variants) is that there are many ways of doing the same thing. If one way works better for your situation, use it. You can write rigid and bulky shell scripts, efficient ones, elegant ones, or anything in between. Make sure you comment the elegant solutions for the next maintainer of your code.

Creating a cron job

This idea of using a file for yesterday's date was a good one. This got a thought process going for us. What about today's date and tomorrow's date, too? A much more efficient and elegant way to get yesterday and today is to create a cron job to run at 12:01 A.M. with the following code:

```
cp /tmp/today /tmp/yesterday
date +%m/+%d/+%Y > /tmp/today
```

You can string both of these commands together on the same cron line with a semicolon between them (for example, `.../tmp/yesterday; date +%m...`) or you can put them in their own script file and run from the cron line. We prefer the former so that the script isn't run accidentally at the wrong time, which would mess everything up, requiring a manual fix.

If you need to consider five-day workweeks rather than seven, then run the cron job Monday through Friday (1-5) instead of all seven days (*). After 12:01 A.M. on Monday, /tmp/yesterday will contain the previous Friday's date. However, beware not to try using this for a Saturday or Sunday script. On the weekend, /tmp/yesterday will contain the previous Thursday's date and /tmp/today will contain Friday's date. Don't forget to change it to a seven-day job if you have weekend jobs that use these dates.

To get started, create /tmp/today manually by either using the second line of the code or with your favorite editor. Thereafter, it will feed itself.

Tomorrow

Okay, we have yesterday and today, but what about tomorrow? Well, don't throw away that code from July's issue just yet. We'll leave it as an exercise for you to modify the previous yesterday code and turn it into tomorrow. This and other articles are available on the Web at www.zdjournals.com/sun/.

Click on Back Issues and find July 1999. The code, along with other article's code, is available for download from the archives at ftp://ftp.zdjournals.com/sun/.

Click on yesterday.sh. Just add one to the date and check for the last day of the month for your rollover, rather than the first.

As always, if you have suggestions, questions or comments about any articles or tips, feel free to contact us at sun@zdjournals.com. That's how improvements are made.

Choosing a good password with npasswd

by Boris Loza

If a computer system's security is one of your concerns, you need not be reminded that one of the first points of attack from hackers is the user's password. Even though you may think you have taught users how to select good passwords, you can still come across many guessed passwords after running some password strength checking software.

Each time we've checked our user's accounts, we've come up with as many as 10 percent easily-cracked passwords. This was a real nightmare until we decided to find a way to prevent users from using easy-to-guess passwords.

Avoiding easy-to-guess passwords

Unfortunately, SunOS doesn't supply system utilities that restrain the ways users select their passwords. What we needed was a utility that would help our users to easily choose a secure password.

We have found several utilities that can do this job for us: passwd+, anlpasswd, password coach, obvious-pw, npasswd, genpassword, and cracklib. Development and technical support for some of these utilities have been discontinued, and some are too old or not well documented.

Taking all this into consideration, we decided to try npasswd, developed by Clyde Hoover from the University of Texas at Austin. Various people assisted in its development. Alec Muffet, for example, provided npasswd developers with his famous Crack's "modules ready to use and a copious collection of word lists."

Npasswd replaces the passwd utility for UNIX. You can download its latest release (version 2.05) from

www.utexas.edu/cc/unix/software/npasswd/download1.html. You'll find documentation at **www.utexas.edu/cc/unix/software/npasswd/doc/**, or in the doc directory in the source distribution, or in <your_install_path>/lib/passwd/ doc when the installation is complete.

Guessability tests

Hoover designed npasswrd to provide a series of guessability tests. The user's password is accepted only when it passes all of the tests. You can customize which tests npasswrd uses and in what order these tests should be applied with the configuration file. Some tests are mandatory, and others optional. The password tests are History and Lexical.

History

This test is optional. Password candidates are compared to the passwords in the user's history and rejected if found. When a password change is done, the new encrypted password is stored in the history database.

Lexical

This test is mandatory. The lexical check includes:

- Enforcing a minimum length of six characters.
- Checking for non-printable or forbidden characters.
- Denying excessive adjacent repeated characters.
- Encouraging a diversity of character classes (mixed case, numbers, punctuation).
- Looking for easily guessed patterns (Social Security numbers, telephone numbers, etc.).

Criteria that can be modified

The following criteria can be modified in the configuration file:

- **Local** (optional). This is the hook for the insertion of site-specific tests. The standard code checks if the candidate is a variation of the system hostname, aliases, or entries in the user's .rhosts file.
- **Password** (mandatory). The candidate is examined to see if it is derived from the user's previous password information.
- **Dictionary** (mandatory). Npasswd checks the candidate against words in various dictionaries. The candidate is rejected if it can be derived from any word in the dictionaries. A variant of the Crack password guessing code is used for this purpose.

Installing npasswd

To install npasswd, run `gunzip` and `tar xvf` on the npasswd-2_05_tar.tar file. Then, change to the directory npasswd-2.05 and run Configure—the script to set the options to be incorporated in npasswd. The mining of the various options can be found in the npasswd documentation. You can safely hit [Enter] on most of the default options. Answer *n* for the Replace System Programs? prompt. We'll explain why later.

In our case, we chose to install npasswd under the /usr/local/ directory. This path would be our relative path to all npasswd files. After Configure was complete, we typed *make*, and then logged on as root and typed *make install*. Note that the installation process doesn't create the password's history database. Use the history_admin utility to create and manage the database. That's it! You'll find the npasswd object file under the /usr/local/lib/passwd/ directory.

Editing the passwd.conf file

Now let's edit a passwd.conf file. This file is used to tell the npasswd what to allow and what to check for in a new user password. You can modify this file to meet your own security policy. For example, adding the following lines:

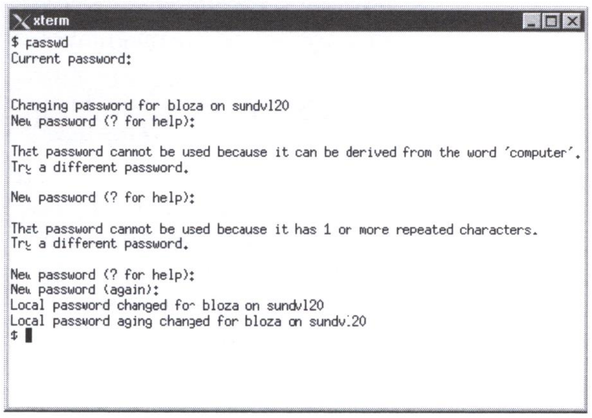
```
Passwd.MinPassword        8
passwd.CharClasses        3
passwd.History depth      5
passwd.MaxRepeat          1
```

tells npasswd that the user password must be at least eight characters long, checks for at least three different character classes in user passwords, remembers the last five passwords that a user employed, and allows only one repeated character. For all npasswd options, see **Table A**. You'll also find a complete list of the configuration options in the passwd.conf file itself.

If you elect to replace the system passwd binary with npasswd during installation, it copies instead the /usr/bin/passwd and puts an original passwd utility under the /usr/local/lib/passwd/system/ directory.

We decided not to replace the system /usr/bin/passwd file with the npasswd binary (that's why we answered no to the prompt asking if we wanted to replace system programs) for several reasons. First, npasswd doesn't allow you to delete a user password and force a user to choose a new one on the next login. You must specify a dummy password and notify a user what the password is. You also have to use `passwd -s -a` instead of the `-sa` (show attributes for all entries) option. When you try to use `passwd -e` (change user shell) and `passwd -g` (change user GECOS information), npasswd misbehaves and the session needs to be restarted. Also, patching the /usr/bin/passwd may replace the npasswd binary.

So, we just linked /usr/local/lib/passwd/npasswd with /usr/local/bin/passwd and put this before /usr/bin/ in the `PATH` variable in the user profile. So, when the user types a passwd command, he'll start npasswd. In Figure A, we've depicted a typical password



```
$ passwd
Current password:


Changing password for bloza on sundvl20
New password (? for help):

That password cannot be used because it can be derived from the word 'computer'.
Try a different password.

New password (? for help):

That password cannot be used because it has 1 or more repeated characters.
Try a different password.

New password (? for help):
New password (again):
Local password changed for bloza on sundvl20
Local password aging changed for bloza on sundvl20
$ █
```

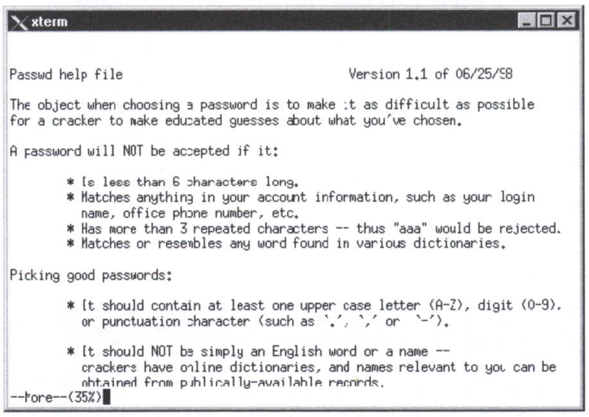
Figure A: *This is a typical password change session.*

Table A: *Configuration options for the passwd.conf file*

Directive	Value	Description
PasswdTolerance	number	Tolerance between old and new passwd files
ShadowTolerance	number	Tolerance between old and new shadow files
MatchTries	number	Chance to give user to correctly enter a password
MatchWait	number	Delay after the user enters an incorrect password
passwd.AlphaOnly	boolean	Allow alpha-only passwords
passwd.CharClasses	number	Minimum number of character classes required
passwd.Dictionaries	path	Password check dictionaries location
passwd.DisallowedChars	string	Change list of non-allowed characters
passwd.Help	path	Help file for passwd
passwd.History age	number	Password history age
passwd.History dbm	path	Set path to history db (DBM database)
passwd.History depth	number	How many passwords to keep per user
passwd.History file	path	Set path to history db (flat file)
passwd.History	none	Disable password history
passwd.LengthWarn	boolean	Warn about passwords over MaxPassword length
passwd.MaxPassword	number	Maximum effective password length
passwd.MaxRepeat	number	How many repeated characters allowed
passwd.Message	path	Message of the day file for passwd
passwd.MinPassword	number	Minimum password length
passwd.PasswordChecks	string	Choose password check functions
passwd.PrintableOnly	boolean	Deny non-printable characters in passwords

change session. By typing a question mark (?), a user can see Help, as shown in **Figure B**.

You can modify the Help contents by editing the /usr/local/lib/passwd/passwd.help file. Or you can create a message of the day for npasswd by altering a passwd.motd file. If you don't replace the passwd with the npasswd, the latter isn't invoked when the system password aging forces the user to change his password (or `passwd -f login_name` is issued). Because we didn't replace the passwd binary with the npasswd, we decided to implement our own method of password expiration.



```
Passwd help file                                     Version 1.1 of 06/25/98

The object when choosing a password is to make it as difficult as possible
for a cracker to make educated guesses about what you've chosen.

A password will NOT be accepted if it:

        * Is less than 6 characters long.
        * Matches anything in your account information, such as your login
          name, office phone number, etc.
        * Has more than 3 repeated characters -- thus "aaa" would be rejected.
        * Matches or resembles any word found in various dictionaries.

Picking good passwords:

        * It should contain at least one upper case letter (A-Z), digit (0-9),
          or punctuation character (such as `.', `,' or  `-').

        * It should NOT be simply an English word or a name --
          crackers have online dictionaries, and names relevant to you can be
          obtained from publically-available records.
--More--(35%)
```

Figure B: *A user can read Help by simply typing a question mark.*

Implementing your own method of password expiration

First we disabled the password aging by typing *passwd -x -1 user_name*. To do this, you have to be a root. Next we needed to force the users to change their passwords every 60 days or any time we want them to do so (like using the `passwd -f` command).

We achieved this goal in two stages. First we developed a shell script to create a marker file .newpasswd.<user_name> in the user home directory, as seen in **Listing A**. This marker file is checked when the user logs on. If this file is found, the user will be forced to change their password.

We've altered the user's profile in order to call the script that looks for the marker file, as seen in **Listing B**. The marker file is unique to each login name, so users can share the same home directory.

This script logs successful password changes to the local3.notice facility of syslogd, which writes to the /var/adm/npasswd file. To do this, we reconfigured the /etc/syslog.conf file. We forced users to change their

Listing A: *Script to create a marker file*

```
#!/bin/sh
#
nawk '
BEGIN {
   FS=":"
}

# Ignore system default users.
# Add more users here if you do not want them to be
# forced to change their password.
$1 == "root"    {next}
$1 == "daemon"   {next}
$1 == "bin"      {next}
$1 == "sys"      {next}
$1 == "adm"      {next}
$1 == "lp"       {next}
$1 == "smtp"     {next}
$1 == "uucp"     {next}
$1 == "nuucp"    {next}
$1 == "listen"   {next}
$1 == "nobody"   {next}
$1 == "noaccess" {next}
$1 == "nobody4"  {next}

# Ignore users with / as a home directory (do not want
# to stomp on root).
$6 == "/" { next }

# For everyone else, create a marker file in their
# home directory.
{
   # Create marker file in users home directory
   marker_file=$6"/.newpasswd."$1

   if(system("/usr/bin/test -d " $6))
      print "Warning: ",$1,"has no home directory",$6
   else
      if(system("/usr/bin/touch " marker_file " 2>/dev/null"))
         print "Warning:  Could not create marker file for",$6
      else
         print "Created marker file for",$1
} ' /etc/passwd
```

passwords every 60 days by adding one line into the root's crontab. Forcing users to change their passwords on the next login can be done by modifying the .newpasswd. <user_name> file in the user home directory.

Checkpassword

Another useful utility that npasswd comes with is the `checkpassword` command. You can do a preliminary check of a password without changing it. Just type *checkpassword*, and at the prompt type a password you'd like to check (for more information see the man pages for checkpassword).

We also created our own password dictionaries that contain our company and business-related words. The password dictionary is a text file with one word per line. Applying the `makedict` command creates the dictionary hash files with the suffixes .pwd, .pwi, and .hwm. To make a dictionary accessible to npasswd, copy the hash files (or make symbolic links) into the appropriate directory. The files should be world-readable. Npasswd will reject any dictionary with a world-writable hash file. For more information on how to create a dictionary file, you can read the online documentation.

Removing npasswd

Removing npasswd is an easy task. If you decide not to use npasswd anymore, you have to edit the user profile and put the /usr/bin/ before /usr/local/bin in the `PATH` variable. Disable a crontab entry for the npasswd 'aging'. Enable password expiration by typing as root

```
passwd -n A -x B <user_name>
```

where `A` is the minimum days before the password can be changed and `B` is the maximum number of days.

Listing B: *Script to look for a marker file in the user home directory*

```
#!/bin/sh

USERNAME=`who am i ¦ cut -d" " -f1`
MARKER_FILE=$HOME/.newpasswd.${USERNAME}
LOGGER="/usr/bin/logger -t PASSWDCHG -p local3.notice"
PASSWD=/usr/local/bin/passwd

trap 'exit' 1 2 3 5 15

if [ -f ${MARKER_FILE} ]; then
{
   echo
   echo
   echo "******************************************************"
   echo "Please change your password now."
   echo "You will not be allowed system access until you do!"
   echo "******************************************************"
   echo
   echo
   ${PASSWD}

   if [ $? -ne 0 ]; then
      exit 1
   fi

   ${LOGGER} ${USERNAME} has successfully changed password
   rm -f ${MARKER_FILE}
}
fi

#Put traps back to normal

trap - 1 2 3 5 15
```

Conclusion

After enabling npasswd, our lives became easier. You just need to periodically maintain a history database and check the /var/adm/npasswd file for users who don't change their passwords on time.

Solaris Access Control Lists

by Edgar Danielyan

As all UNIX variants, Solaris has the traditional UNIX concept of file access permissions (where permissions may be set on the basis of user, group, and others). But Solaris also has an advanced ACL (Access Control List) system that can give you even greater control over your files.

While most of the time the good old access permissions concept is all that we need, an ACL may be very useful in certain circumstances, such as when it's necessary to make exceptions to the access permissions. ACLs provide the possibility of setting access permissions on a per-user and per-group basis, thus providing total control over the file or directory. To find out whether a file has an ACL, use the `ls -l` command:

```
# ls -l hello.c
-rw-------+  1 edd   people     267
➥Dec 1 13:13 hello.c
```

A plus sign (+) after access permissions says that hello.c has an ACL set. It's possible to set an ACL on a per-directory basis. In this case, all newly created files will have the default ACL, which is set on the directory.

Displaying an ACL

To display ACL information for a particular file, use the following `getfacl` command:

```
# getfacl hello.c

# file: hello.c
# owner: edd
# group: edd
user::rw-
user:inna:r--                 #effective:r--
user:ran:rw-                  #effective:rw-
group::---                    #effective:---
mask:rw-
other:---
```

Table A: *The various setfacl parameter descriptions*

Parameter	Description
user::perms	Set owner permissions
group::perms	Set group permissions
other::perms	Set permissions for others
mask::perms	Set ACL permissions mask (maximum permissions allowed)
user:uid:perms	UID may be either user name or numeric UID
group:gid:perms	GID may be either group name or numeric GID
default:user:perms	Default user permissions
default:group:perms	Default group permissions
default:other:perms	Default permissions for others
default:mask:perms	Default mask
default:user:uid:perms	Default permissions for specified user
default:group:gid:perms	Default permissions for specified group

If more than one filename is supplied, the output information is separated by a blank line. There are two options to `getfacl`. `-a` displays the filename, file owner, file group, and ACL entries for the file or directory. `-d` displays the filename, file owner, file group, and the default ACL entries for the directory.

Setting ACL

To set up an ACL for a file or directory, use the following `setfacl -s` command:

```
# setfacl -s user::rw-,group::---,
➥other:---,mask:rw-,user:inna:r--,
➥user:ran:rw- hello.c
```

Table A shows the descriptions for the various options. Note that one or more ACL entries are separated by commas. Please note that if the file already has an ACL, it will be replaced with the specified one

Modifying and deleting an ACL

The command for ACL modification is the same `setfacl`, but with a different option, `-m`:

```
# setfacl -m user:ran:rw- hello.c
```

Unlike the previous `setfacl`, this one doesn't replace the old ACL, but modifies the existing entry (in this case, for the user ran). `setfacl -d` is used to delete the ACL for a file:

```
# setfacl -d user:inna hello.c
```

Copying an ACL

It's possible to copy the ACL just by redirecting the output of the `getfacl` command to the input of the `setfacl` command:

```
# getfacl hello.c | setfacl -f - bye.c
```

Notes

We have one word of caution: when ACLs are used incorrectly, they may cause lots of (difficult to catch) problems. Therefore, before deciding to use ACLs, we advise clearly defining the access policy and double-checking it before implementing ACLs. Also you should note that ACLs aren't available on all file systems.

Introducing the Internet Message Access Protocol

by Edgar Danielyan

Nowadays, most of us use POP3 (Post Office Protocol, version 3) to fetch and read our email from the mail server. While POP may be considered adequate for the majority of email users, there's an alternative to POP, which has more advantages and less inconveniences. It's the Internet Message Access Protocol (IMAP), described in RFC 1725 and 1730. A brief comparison will reveal that IMAP could be a much better alternative, or at least an addition, to current POP3 service.

Features common to both POP3 and IMAP

These are the characteristics common to both POP and IMAP:

- Both POP and IMAP support offline operation.
- Mail is delivered to and stored on a dedicated mail server.
- Mail may be accessed from any node on the network.

- Both protocols are open and standardized.
- There are freely available implementations of both protocols for almost all operating systems.
- Clients are available on all PC platforms.
- There are a number of good quality commercial offerings.
- Both protocols rely on SMTP (Simple Mail Transfer Protocol) to send email.

Differences

Now, let's take a look at POP's and IMAP's differences.

POP

- Relatively simple to implement protocol
- Lots of available software, both on client and server
- Has only offline mode

IMAP

- Relatively complex protocol—not as easy to implement as POP
- Less available software than POP
- Has message status flags—that is, messages in mailboxes may be marked as deleted, answered, fetched, high priority, etc.
- Is extensible (that is, new features may be added)—IMAP has a notion of extension, that's an additional feature added by a vendor or newer specification of the protocol
- Can store (in addition to fetching) messages in remote mailboxes
- Supports multiple mailboxes on remote server—user can remotely create, delete, move, and rename mailboxes on the server
- Optimized for low-bandwidth links—no need to download all messages; can selectively download the whole message or part of it (if the message is in MIME format)
- Can be used to access non-mail data (that is, news, documents, etc.) on the server
- Support for shared mailboxes—single mailbox may be accessed by many users for information sharing
- May be used to read email from different computers—without downloading the whole mailbox
- Works in both online and offline modes

IMAP's advantages

To summarize IMAP's advantages over POP: much more features and much less restrictions. To make things even better, there's a freely available source-code implementation of IMAP version 4 revision 1 server for UNIX systems written by Mark Crispin at the University of Washington. You can download it from ftp://ftp.cac.washington.edu/mail/imap.tar.Z.

It compiles and runs on the following systems: Solaris, AIX, AmigaDOS, AOS, Altos, A/UX, FreeBSD, NetBSD, OpenBSD, ConvexOS, DG/UX, Bull, Dynix, EP/IX, HP-UX, Ultrix, Linux, LynxOS, MachTen, NEXTSTEP, OSF/1, PTX, Pyramid, QNX, SCO, Irix, Sinix, and UnixWare.

Compiling and installing UW IMAP server on Solaris systems

After unpacking the distribution, take a look at the Makefile—there are a number of options that you may want to customize—although default options should work on the majority of systems. Depending on the C compiler installed, make gso on system with a GCC compiler, or make sol on system with the standard Sun C compiler.

Assuming everything goes well, at the end of the make process you'll end up with an imapd in imapd, and ipop3d in ipopd (we highly recommend using the bundled POP3 server [ipop3d] with the imapd). Strip the binaries using /usr/ccs/bin/strip and install them in your chosen directory (/usr/local/bin would be fine). Now it's necessary to add IMAP entries to both /etc/services and /etc/inetd.conf. Add the following line to /etc/services:

```
imap            143/tcp
```

This declares a TCP protocol called IMAP, on 143rd TCP port. To configure inetd to run imapd for incoming IMAP connections add the following line to /etc/inetd.conf:

```
imap    stream  tcp     nowait  root
➥/usr/local/bin/imapd    imapd
```

This defines IMAP as a stream-oriented, TCP protocol, and tells the inetd to run /usr/local/bin/imapd as root each time a connection is made to the IMAP port (TCP 143). Restart the inetd using kill—HUP <pid of intetd> and voila, you're running an IMAP server! Now it's time to configure your email client to use IMAP. For Netscape Messenger, you only need to change Protocol in the Mail servers from POP3 to IMAP and set your preferences.

Multi-user cron

by Paul A. Watters

I want to set jobs to run automatically at certain times of the day on my Sparc, but I can't seem to get crontabs working for different users. How do I go about it?

Writing and submitting cron jobs for different administrative users is a common enough task for root (privileged) users; however, there are a number of issues which users should be aware of. The first is that we don't recommend that users edit their crontab files directly. Instead, the command `crontab -e` should be used, which invokes the default editor, and which resubmits the job to the cron daemon after changes have been saved. Simply editing a crontab file doesn't resubmit a user's entry, but the cron daemon can be restarted by finding the crond process

```
ps -eaf | grep crond
```

and issuing a `kill -1` signal to that process.

The second issue to be mindful of is the default editor, which is invoked when a user issues the `crontab -e` command. Unless the environment variable `EDITOR` is explicitly set within the user's shell, for example, with the command

```
EDITOR=vi; export EDITOR
```

in Bourne shell, the default UNIX editor `ed` is invoked. If this is invoked in error, and the user isn't familiar with `ed` commands, the editor can be exited by issuing an EOT (that is, in the form of [Ctrl]D).

The final, and most important, issue is that the root user should *never* edit another user's crontab file. As the man page for crontab(1) states, the "resulting behavior may be unpredictable." One example of this unpredictable behavior might be that the user's crontab file, which root is editing, will be written as root's crontab file. This means that precious information about root's cron jobs can be easily lost. We recommend (which is much safer) that the root user use `su` to edit crontabs for different users. For example

```
# su - nobody -c "crontab -e"
```

will edit and submit nobody's crontab file safely.

Command line user maintenance

I currently use admintool to add, modify, and delete users on my system. Is there a faster way to do this on the command-line?

Yes, there are much faster ways to add and delete users from the system. It's possible, for example, to process information from the password file /etc/passwd directly in a shell script, writing a new password file to an alternative file (one should never operate directly on /etc/passwd). For example, to get a listing of all users of the dba group with gid 501, you can write a script using a grep of the form

```
grep 501 /etc/passwd
```

To add users, there's a useful command-line program called useradd, which has the form

```
useradd -c comment -d dir -g
➥group -u uid -s shell username
```

There are other options available, but these are the most commonly used. For example, if I want to add a user pwatters with the comment Watters_P, home directory /staff/pwatters, group 101, and uid 101, I would use the command

```
useradd -c  Watters_P -d /staff/pwatters
➥-g 101 -u 101 -s /bin/sh pwatters
```

Upgrade blues

I finally upgraded my Sparc 20 from Solaris 1.1.3 to Solaris 7 due to Y2K concerns. However, I tried to build my first GNU package, and I received the error message:

```
"configure error: no acceptable cc
➥found in $PATH".
```

Where has my C compiler gone?

This is the most frequently asked question on comp.unix.solaris and other discussion forums for Solaris. There are a number of significant changes between Solaris 1.x and Solaris 2.x, one of them being that Sun no longer bundles the operating system with a C compiler, even though there might be a file called /usr/ucb/cc.

You will either have to purchase one of Sun's commercial C compilers or use the ANSI compliant GNU C compiler (gcc), which is widely used in industry. Of course, there is a chicken and egg problem here: how can you compile a compiler from the Free Software Foundation if you don't have a compiler to start with? Fortunately, there's a fantastic resource called Sun Freeware found at **www.sunfreeware.com**). Sun Freeware has binary distributions of the GNU C and C++ compilers in Solaris package format, so that they can be easily added to the system with a command like

```
pkgadd -d solaris-gcc-2.8.1.local
```

Of course, after you've installed the compiler in this way, we advise testing your installation by obtaining the sources for the gcc compiler and recompiling it with itself. The Sun Freeware site also has binary distributions for most Solaris freeware and GNU packages, and is an invaluable resource for those who have no qualms about downloading and installing binary distributions from untrusted sites.

Hunting for text

A friend just emailed some Microsoft Word files to me, and I urgently need to see their contents. I don't have access to an X-terminal or graphics console. Is there any way that I can extract the text? Also, how do I look at what's happening behind the scenes when Solaris is executing a command like string?

Fortunately, there's a command in Solaris called `strings` that allows the user to extract printable strings from a binary file (like Microsoft Word files). Although the formatting would obviously be lost, this is a useful way to extract text from binary document files.

The easiest way to trace system calls and program execution is by using the `truss` utility. This will show all the system calls being executed in context, and in sequential order. For example, when a file's text contents are displayed using strings, the `truss` command is likely to display a lot of lines like those shown in **Listing A**.

These system calls are reasonably straightforward to interpret, and are very useful when trying to determine why a program isn't working in the way that you expect. (For example, when a file isn't found, that failure is explicitly reflected in error codes from a file read call.)

Listing A: *Truss shows us system calls*

```
write(1, " Hello World!\n", 12)                = 12
write(1, " The quick brown fox".., 13)         = 13
write(1, " jumps over the lazy dog.".., 16)    = 16
read(0, 0x000224BC, 8192)                      = 0
llseek(0, 0, SEEK_CUR)                         = 60928
_exit(0)
```

About our contributors

Clayton E. Crooks II is a self-employed computer consultant living in Knoxville, TN. He's married and has one child. Clayton's hobbies include game development, 3D modeling, and any athletic activity he can find time for. He can be reached via email at crooks@planetc.com.

Edgar Danielyan is currently working as a network administrator and manager of a top level domain of Armenia. Previously, he spent some time studying US and UK law. He's also worked for the United Nations, the ministry of defense, a national telco, a bank, and has been a partner in a law firm. He speaks four languages, likes good tea, and is a member of ACM, IEEE CS, SENIX, CIPS, ISOC, IPG, and many other much less known organizations. He can be reached at edd@computer.org.

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.

Boris Loza holds a Ph.D. in computer science from Russia. He worked as a UNIX administrator and developer for 10 years. Currently he's working for Fidelity Investments Canada in the position of Data Security and Capacity Planner, doing IT security for UNIX, Windows NT and Novell. He has a daughter Anna and likes reading computer and mystery books and watching movies. He can be reached at Boris.Loza@FMR.com.

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
Managing Editor	Joe Froehlich
Copy Editors	Rachel Krayer, Christy Flanders, Ellyn McCasland
Contributing Editors	Clayton E. Crooks II, Edgar Danielyan, Jeff Forsythe, Boris Loza, Paul A. Watters
Print Designer	Melissa Ribaudo

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@zdjournals.com.

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

ZD Journals

General Manager	Kelly Baptiste
Manager of Customer Relations	Heather Loaicono
Manager of Operations	Cristal Haygood
Manager of Print Design	Charles V. Buechel
Manager of Product Marketing	Mike Mayfield
Product Marketing Manager	Brian Cardona

Postmaster

Periodicals postage paid in Rochester, NY and additional mailing offices.

Postmaster: Send address changes to

Inside Solaris
P.O. Box 92880
Rochester, NY 14692

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 a back issue from the last six months, 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. Collections of back issues are available on CD-ROM as well. Please call for more information.

Are you moving?

If you've moved recently or you're planning to move, you can guarantee uninterrupted service on your subscription by calling us at (800) 223-8720 and giving us your new address. Or you can fax us your label with the appropriate changes at (716) 214-2386. Our Customer Relations department is also available via email at zdjcr@zd.com.

2096

USPS ARMIN PS1 881 APPROVED POLY

Sun Technical Support
(800) 786-7638

PERIODICALS MAIL

*************3-DIGIT 480
C: 7661905 00002096 04/00 17 28
[redacted]

Please include account number from label with any correspondence.

NET.UPDATE

News around the 'net

by Edgar Danielyan

Spam is illegal in Ontario

The Superior Court of Ontario ruled that spamming violates netiquette and, in one particular case, the usage policy of the ISP which was used by spammer.

U.S. Critical Infrastructure Council

Late this summer, President Clinton signed an executive order creating the U.S. Critical Infrastructure Council, which is to include representatives from all law enforcement agencies and representatives from the private sector. One of the goals of the new Council is to protect the United States from cyberterrorism.

Digital signatures are legally binding in California

Earlier this year, the Governor of the State of California signed into law a bill that made digital signatures legally binding and equivalent to signatures on paper.

Electronic signatures are used by the U. S. government

Two U.S. congressmen digitally signed the Year 2000 Readiness and Responsibility Act and sent it via encrypted email to the President. It was the first time in history that digital signing was used in the U.S. Congress.

Accreditation Program for IT Laboratories

U.S. National Institute of Standards and Technology (NIST) announced an official accreditation program for organizations that test commercial information security systems for compliance with the U.S. and international standards.

Security spending to increase in 2000

The U.S. Senate and the House of Representatives have prepared bills to increase security funding for the Department of Defense and the Department of Energy in 2000. Certain differences that exist between these two bills will be settled later.

Internet is *not* for everyone, regretfully

A recent Internet access survey published by the Communications of shows that Internet access costs in Armenia (adjusted to percent of GDP) are 485 times more expensive than in Finland:

Armenia	$121
Kenya	$100
India	$82
Argentina	$50
Ghana	$50
Dominican Rep	$26
USA	$20
Finland	$18

Such costs are partly due to the high cost of international data circuits (per month per 64k):

Kenya	$11,000
Armenia	$9,000
Nigeria	$6,000
Argentina	$4,800
Ghana	$2,500
USA	$300