Volume 7, Number 3 • March 2001
www.elementkjournals.com/sun
U.S. $11.00

Inside Solaris™
Tips & Techniques for users of Sun Solaris

Role-based Access Control

by Alan Orndorff

Operating System: Solaris 8

UNIX has always allowed users to be granted access privileges for running programs that they normally don't have access to. Even with modern-day Solaris, you still have the ability to set your user id and group id, as well as access control lists, and you can still use the venerable third-party sudo program.

Things have changed just a tad with Solaris 8, though. We now have a process known as *Role-based Access Control* (RBAC). Although sudo is a bit easier to set up, with RBAC you no longer need to compile another program to provide this functionality.

Foundation

There are essentially two directories and four files that are used to configure RBAC. The file in the /etc directory is user_attr. The files in the /etc/security directory are auth_attr, prof_attr and exec_attr.

The */etc/user_attr* file is an extension to /etc/passwd and /etc/shadow. When you create a role, it's stored in this file.

A role is similar to a typical user account, except you can't log on to the system with a role user account. You log on as a normal user and then su to the role. The role determines which commands your users have access to outside of the normal user commands.

The */etc/security/auth_attr* file is used to grant access to normal system-type commands, such as changing the date and time or Printer Management. When defining a custom role, you may or may not need to update this file.

The */etc/security/prof_attr* file contains execution profiles. A tag in this file matches tags in the exec_attr file.

The */etc/security/exec_attr* file is where the actual commands that the user can run are stored. This is the key to the whole system.

All of these files use the following delimiters:

- **Colon.** Used as a field separator within the file. For example, you could have a line that looks as follows:

```
Fruit: Vegetables: Meats: Breads
```

- **Semicolon.** Used to separate key value pairs within the file. We could separate apples and lemons like so:

```
apples=red; lemons=yellow
```

- **Comma.** Used to separate an ordered list within the file:

```
lions, tigers, bears
```

- **Period.** Used to separate the prefix from the suffix. This separation defines execution profiles with finer granularity:

```
solaris.system.date
solaris.system.shutdown
```

To create a role-based access control, you add the role to user_attr and then create the entries in prof_attr and exec_attr.

In This Issue

element k JOURNALS
A Division of Element K Press

You can use the built-in authority schemes as defined in the auth_attr file, or you can add your own to this file as well.

The four files

It's important to understand these files and how they relate to each other. To help you understand them, we'll now describe the use of each file in detail.

/etc/user_attr

The first file we'll examine is /etc/user_attr. The Extender User Attributes (/etc/user_attr) file is an extension to your normal /etc/passwd and /etc/shadow files. This file allows you to assign roles to users, specify which account you'll use to run certain commands, and assign users to profiles and authorizations.

The role account also has special shells associated with it. Instead of assigning the typical shells such as bourne, csh, ksh, etc., you select one of the three profile shells:

- **pfsh.** Profile bourne shell.
- **pfcsh.** Profile C shell.
- **pfksh.** Profile Korn shell.

If the user decides to change the shell to any shell other than these, then the role commands will *not* work. You must be in one of these special shells to execute the role commands.

The /etc/user_attr file has the following fields:

- **User.** The name of the user or role as it appears in /etc/passwd.
- **Qualifier.** Reserved for future use.
- **Res1.** Reserved for future use.
- **Res2.** Reserved for future use.
- **Attr.** An optional list that describes the security attributes to be applied when the user runs the commands. There are four valid keys:
  - **Auths.** Specifies a name chosen from the names defined in /etc/security/auth_attr. These names can contain wildcards.
  - **Profiles.** Specifies a name chosen from the profile names specified in /etc/security /prof_attr. A profile determines which commands a user can execute. There are some severe security implications for Profiles that we'll discuss later.
  - **Roles.** Defines each role needed. They are defined in this same file. Roles are defined by setting the type to role. You can't assign roles to other roles.
  - **Type.** Indicates what kind of account it is. Type can be set to "normal," which indicates that it's a normal user, or to "role" if the account is for a role.

These are all of the possible elements that the user_attr file can contain. The following is an example:

```
root::::type=normal;auths=solaris.*, \
solaris.grant;profiles=All
```

The first field specifies root. We know that it's a user account, as the type is set to "normal." The root user account is authorized to use the solaris.* and solaris.grant authorizations as defined in auth_attr, and can utilize commands that are defined for the "All" profile.

An example of a role entry would look like this:

```
filemngr::::type=role;profile=Filesystem
➥Management,All
```

The name of the role is filemngr, as indicated by the type=role tag, and the role has access to the commands that are allowed by the "Filesystem Management" profile and the "All" profiles.

/etc/security/auth_attr

The second file we'll look at is /etc/security /aut_attr. This file contains authorizations to restricted system functions. Some of these functions include changing date/time, system shutdown, auditing, and other system-type functions. You can create your own system definitions and include them in this file as well. Any function that starts with "solaris.*something*" is defined by Sun. If you want to add your own authorizations, we recommend that you use your company Web address in reverse format (i.e., com.yourcompany .authorization_name). By doing this, you'll ensure that future upgrades or changes in future versions of Solaris won't wipe out your custom definitions.

You can add authorizations for users or roles in one of two ways. You can either add authorizations directly in the /etc/user_attr file, or indirectly by adding them to profiles and then assigning the profile to a user or role.

Certain privileged programs use this file to determine the users' rights to execute that program.

This file also defines the location of the Help file to use in GUI programs. The file has the following fields:

- **Authname.** A unique string used to define the authorization in the format of prefix.suffix. The prefix should be your company's domain name in reverse syntax, and the suffix should be the functional area that you're granting authorizations for. This might look something like com .yourcompanyname.authname. If the authname ends in a period, then it doesn't grant or deny access to programs, but can be used within a GUI application as a heading.
- **Res1.** Reserved for future use.
- **Res2.** Reserved for future use.
- **Short-desc.** A short description suitable for displaying in a GUI.
- **Long-desc.** A long description that could be used in a GUI Help file.
- **Attr.** An optional list that describes the attributes of the authorization. The Help keyword identifies a Help file in HTML format. You can access Help files from the index.html file in the /usr/lib/help/auths/locale/C directory.

Some example entries in this file would look like:

```
solaris.jobs.:::Cron and At Jobs::
➥help=JobHeader.html
solaris.jobs.admin:::Cron & At
➥Administrator::help=JobsAdmin.html
solaris.jobs.grant:::Delegate Cron &
➥At Administration::help=JobsGrant.html
```

The first entry ends in a period, so its use is limited to a heading in a GUI application. The second entry allows a user to view or change cron or at jobs on the local machine. The third entry is also a special type. Whenever an authorization ends in .grant, it allows the user to grant authorizations to other uses. Be careful of using .grant, as whichever user you give this right to could arbitrarily let other users use these commands as well.

/etc/security/prof_attr

An execution profile is a way of bundling together all the authorizations and commands that you then assign to users and roles.

The file has the following fields:

- **Profname.** The case-sensitive name of the profile.
- **Res1.** Reserved for future use.
- **Res2.** Reserved for future use.
- **Desc.** This field should explain why the role exists and why you would let a user run these commands. It should be suitable for use as a Help file for a GUI application.
- **Attr.** An optional list that describes the authorizations or Help files to be used by the command upon execution. The Help file has the same specifications as described for the auth_attr file. The auths keyword corresponds to authorizations created in the auth_attr file. You can specify wildcards here.

A couple of example entries from this file include the following:

```
Audit Control:::Administer the audit
➥subsystem:auths=solaris.audit.config,
➥solaris.jobs.admin;help=AuditControl.html

Device Management:::Control Access to
➥Removable Media:auths=solaris.device.*;
➥help=DevMgmt.html
```

The first example shows that the profile name is "Audit Control." This profile is authorized to run all commands associated with solaris.audit .config in the auth_attr file. The Help file is /usr/lib/help/auths/locale/C/AuditControl.html. The second example shows the use of a wildcard allowing the user to run all commands associated with the solaris.device authorizations as defined in auth_attr.

/etc/security/exec_attr

This file is the key to the whole system. In it, we define exactly which commands the user can run. After you define a policy, you need to edit this file and define which commands that profile is allowed to run, as well as how you want that command to run. In /etc/security/exec_attr, you can tell the system to run the command as user id, effective user id, group id, or effective group id. The file has the following fields:

- **Name.** The name of the profile. This matches the entry in prof_attr.
- **Policy.** The security policy associated with the execution profile. Currently there's only one: suser.
- **Type.** Currently the only valid type is cmd (command).

- **Res1.** Reserved for future use.
- **Res2.** Reserved for future use.
- **Id.** The full path to and the name of the command that the user is allowed to run. You can use wildcards here. If you need to use command line switches, then you'll need to write a script and point the id to the full path name of the script.
- **Attr.** An optional list containing one or more of the following followed by a value: uid, euid, gid or guid. The value or name of the user id, effective user id, group id, or effective user id is the same as defined in /etc/passwd or /etc/group.

A few example entries from this file include:

```
All:suser:cmd:::*:
Audit Control:suser:cmd:::
➥/etc/init.d/audit:euid=0;egid=3
Audit Control:suser:cmd:::
➥/etc/security/bsmconv:uid=0
Audit Control:suser:cmd:::
➥/etc/security/bsmunconv:uid=0
Audit Control:suser:cmd:::
➥/usr/sbin/audit:euid=0
```

The first entry here is of some importance. The "All" profile lets the user run any command on the system without any special privileges. Not granting a user access to the "All" profile takes away this access. The only commands that the user can now run are those that the remaining profiles grant him access to run. By not granting access to the "All" profile, you can severely restrict the commands that the user is allowed to run, even down to just one command.

The rest of the examples show you that the profile "Audit Control" can be set up to run a few different commands in different directories. Also, you can see that some of these commands run as uid=0 or euid=0, and the first command will also run with an egid equal to 3. Make sure that these match up with the permissions of the command on the file system.

Logins and su

In order for the typical user to use these commands, he must log on as himself and then su to the role account. No one is allowed to log on with a role account. A role is created in the same fashion as a user. It has all the attributes that a normal user account has. If you decide that you want to temporarily suspend a role, simply change the password on the role account and don't give anyone the new password. When you're ready to resume the role, either reset the password to what it was or assign a new one and let the users know what it is.

Role commands

The following commands are used to create and manage roles: `roleadd`, `rolemod`, `roledel`, `/etc/init.d/nscd`, `profiles`, `roles`, `useradd`, `userdel`, `usermod` and `vi`. Note that `admintool` isn't listed.

To create a role, use the following `roleadd` command:

```
roleadd -c comment -d home dir -e
➥expire -f inactive -g group -G group
➥-m -u uid -s shell -A  authorization
➥-P profile role
```

Those familiar with `useradd` will have no problem with this command. The `-m` creates the home dir if it doesn't already exist. Make sure that you remember to use pfsh, pfksh or pfcsh for the shell. Before assigning an authorization or profile, they must exist in auth_attr or prof_attr, respectively, before issuing the command. The role account has the same restrictions as a user account in terms of length and other attributes.

To modify a role, you use the `rolemod` command. This command is analogous to the `usermod` command, and follows the same format:

```
rolemod -u uid -g group -G group -d
➥dir -m -s shell -c comment -l new_name
➥-f inactive -e expire -A authorization
➥-P profile role
```

Finally, to remove a role, use the `roledel` command as follows:

```
roledel -r  role
```

As with `userdel`, the `-r` removes the home directory.

If you want to see a list of profiles that you currently belong to, use the `profiles` command:

```
profiles -l user
```

Using the `-l` switch allows you to see which commands are associated with the profile, as well as the special attributes, such as `eid` or `gid`, of the command. If you want to see the profiles assigned to a particular user, then specify the user's name as part of the command.

If you want to see what roles are assigned to your user account, you can use the `roles` command as follows:

```
roles user
```

The `useradd` and `usermod` commands have been updated with additional switches. This includes the `-A` and `-P` options, respectively, which assign authorizations and profiles to the user. They have also added the `-R` option to assign roles to a user.

Example

One of the more useful commands for helping to debug network-related problems is the `snoop` command. Because this command can be used to read passwords, normal users usually don't have access to this command. The following is an example of how to give users access to the `snoop` command. Remember that this is just an example, and you probably will *not* want to do this in the real world.

Add the following line to the end of the /etc /security/exec_attr file:

```
Snoop:suser:cmd:::/usr/sbin/snoop:uid=0
```

Now, update the /etc/security/prof_attr file as follows:

```
Snoop:::Allow users to run the snoop command:
```

Once this is complete, create the following role:

```
roleadd -m -d /export/home/snoop -c
➥"Allow users to run snoop" -s
➥/usr/bin/pfsh -P Snoop,All usrsnoop
```

Then you can set the password for the usrsnoop account:

```
passwd usrsnoop
```

Now, grant the appropriate user accounts the right to use the command:

```
usermod -R usrsnoop user
```

You're now ready to test your setup. Do this with the following commands:

```
su - user
su role (usrsnoop)
run role commands (/usr/sbin/snoop)
```

Inform the end users of the (role) snoopusr account, the password, and what it's used for.

Name server cache daemon

Sometimes when you add or modify a role/profile, the role/profile won't take effect right away. The reason that it may not take effect immediately could be the name server cache daemon. If you make changes and they aren't showing up, you may need to stop and then restart this daemon. To do this, issue the following commands:

```
/etc/init.d/nscd stop
/etc/init.d/nscd start
```

You can also try adjusting your nscd settings by modifying /etc/nscd.conf.

Conclusion

Like anything new, getting used to RBAC can take a little time, but in the long run it will solve problems that are not easily solved by setting UNIX permissions. It also can augment or replace sudo if you're currently using that solution. ✳

Internet protocols: ICMP

by Edgar Danielyan

Operating System: Any that supports TCP/IP

Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP), both described in "Understanding Internet protocols: TCP and UDP," in our February issue, provide transport functions in TCP/IP networks. They provide the backbone of data transport to and from devices and computers. But moving data is not enough to ensure efficient network usage. You handle control and management of TCP/IP networks by the Internet Control Message Protocol (ICMP) and the Simple Network Management Protocol (SNMP). In this article, we'll describe the current ICMP, the next version for IP Version 6, and some extensions to the ICMP.

Internet Control Message Protocol

Let's begin with an explanation of the role of ICMP in TCP/IP networks. As RFC 792 says, "ICMP is actually an integral part of IP, and must be implemented by every IP module." In this regard, Internet Control Message Protocol is different from all other protocols in the IP family in that all devices in an IP network should understand and speak ICMP in order to provide the minimum required control and management functions. Such utilities as ping and traceroute use ICMP to diagnose and troubleshoot IP networks; routers and gateways use ICMP to communicate between themselves and with end nodes, etc.

ICMP messages

ICMP messages are sent in IP packets, where the Protocol field is set to 1 to show that the data part of the packet contains an ICMP message. The first byte of this data part is the ICMP message type number. The following messages are defined in the core ICMP:

Destination unreachable
The destination unreachable message type has the following six subtypes:

- Network unreachable
- Host unreachable
- Protocol unreachable
- Port unreachable
- Fragmentation needed but the Don't Fragment flag is set in IP header
- Source route failed

Time exceeded
This message is sent when the Time To Live (TTL) field in the IP header reaches 0. Subsequently, the packet is dropped and a message is sent back to the sender to notify of the condition.

Parameter problem
When a router or end node receives a packet with incorrect or unknown parameters, it discards the packet. It then notifies the sender that the packet was not delivered.

Source quench
When a gateway or the destination host isn't as fast as the sending host, or if it doesn't have enough buffer space to compensate for the difference in receiving and sending speeds, it may send a source quench message to the sender requesting the rate at which packets are sent to be lowered. Effectively, this is a type of Quality of Service (QoS) feature.

Redirect
In some situations, routers and / or end nodes may redirect traffic. Most of the time, it's being done to increase network efficiency or fix routing issues.

Echo/echo reply
Echo and echo reply messages are used for network diagnostics. The data received in the echo request message must be returned in the echo reply message.

Timestamp/timestamp reply
This is not frequently used. A host may or may not implement timestamp and timestamp reply.

Information request/reply
This message type allows the sender to find out the network number and its IP address. The sender sends a message with source and destination addresses set to null, and the replying party sets them to the actual IP addresses used.

These are the ICMP messages defined in RFC 792. Several new types were added after the original specification was published; we describe some of them in the following sections.

Router discovery messages

RFC 1256, where ICMP Router Discovery extensions are defined, "specifies an extension of the Internet Control Message Protocol (ICMP) to enable hosts attached to multicast or broadcast networks to discover the IP addresses of their neighboring routers." Those of you who have had the pleasure of configuring default routers on dozens (or hundreds) of systems will appreciate this feature. ICMP Router Discovery enables hosts to find out the address of the default router automatically, using two ICMP messages called Router Advertisements and Router Solicitations.

Security failures

ICMP Security Failures Messages are specified in RFC 2521. They are intended to be used with IP Security Architecture (IPsec) Authentication Header (AH) and Encapsulated Security Payload (ESP). Their purpose is to either notify the end node of incorrectly configured IPsec in case of manual configuration, or trigger an automatic reconfiguration when Security Associations (SAs) change.

Domain name messages

ICMP Domain Name Messages are described in RFC 1788, authored by William Allen Simpson. This extension to the ICMP provides a method to find out the fully qualified domain name (FQDN) for a given IP address and addresses the IN-ADDR.ARPA domain issue. With ICMP Domain Name Messages, the IP nodes are directly queried for their FQDNs, thus avoiding all issues associated with the IN-ADDR.ARPA. Unfortunately (in our humble opinion), it isn't widely used.

ICMP Version 6

As its name suggests, Internet Control Message Protocol Version 6 is used in IP Version 6, or IP Next Generation, networks. It is defined in RFC 2463, published in December 1998, and authored by Alex Conta of Lucent Technologies and Stephen Deering of Cisco Systems. As with ICMP Version 4, ICMP Version 6 is an integral part of IP and must be implemented by all IP devices. Essentially, ICMP Version 6 is almost the same protocol as ICMP Version 4, but with some differences. In addition, ICMP Version 6 messages are clearly divided into two groups: error messages and informational messages. The core messages are described in **Table A**. The functions implemented by these messages are almost the same as in ICMP Version 4, but they take into account features and differences of IP Version 6. ✳

Table A: *ICMP Version 6 error and informational messages*

Error messages	Informational messages
Destination Unreachable	Echo Request
Packet Too Big	Echo Reply
Time Exceeded	
Parameter Problem	

References

- RFC 792: Internet Control Message Protocol (ICMP)
- RFC 2463: Internet Control Message Protocol Version 6
- RFC 1256: ICMP Router Discovery Messages
- RFC 2521: ICMP Security Failures Messages
- RFC 1788: ICMP Domain Name Messages

Trusted Solaris 8

by Edgar Danielyan

Operating System: Trusted Solaris 8

Last November, Sun Microsystems released the Trusted Solaris 8 Operating Environment. More than 10 years of research and development culminated in a stable and industry-standards-compliant, trusted system based on widely used Solaris 8. In this article, we'll introduce you to Trusted Solaris 8, the latest trusted operating environment from Sun Microsystems.

What's meant by "Trusted"?

Most of the time when we speak of security, we speak of security from external (to the system or the organization) threats. These external threats are addressed by authentication, encryption, firewalls, etc. However, these measures are unable to effectively protect from insider attacks—unauthorized actions of legitimate users and/or software. Trusted systems in general, and Trusted Solaris in particular, address this issue by providing additional access control, auditing and authorization mechanisms.

Users of Trusted Solaris

Finance, banking, health care, government, military, insurance, and e-commerce are all prime candidates for Trusted Solaris. Any industry requiring high levels of security and fine-tuned access control to data and applications can benefit from the features of Trusted Solaris.

Trusted Solaris features

In addition to all features present in standard Solaris 8, Trusted Solaris 8 has many additional security mechanisms and concepts. We'll briefly describe these in the following sections.

Scalable security

Almost all Solaris applications run under Trusted Solaris 8 without any modifications. Applications developed especially for Solaris 8 enjoy almost absolute compatibility.

Trusted mail

Mail subsystems in Trusted Solaris 8 fully support sensitivity labels and clearance levels. We'll discuss these concepts later.

Name services

Naming services can be another area of vulnerability. Supported naming services in Trusted Solaris include NIS and NIS+, with support for secure remote administration.

Additional Pluggable Authentication Modules

Trusted Solaris 8 includes several additional Pluggable Authentication Modules (PAMs) not found in the standard Solaris 8. These provide a higher level of control over authentication processes in the system.

Device sensitivity levels

Trusted Solaris 8 introduces the concept of sensitivity levels of devices. If a particular device is assigned a sensitivity label, only the owner or users with higher clearance levels may access the device.

Trusted paths

The concept of trusted paths in Trusted Solaris 8 guarantees that the users are indeed interacting with the appropriate software during such important actions as changing passwords or performing system administration. When in GUI mode, the system provides visual indication of the current sensitivity label and clearance level through the Common Desktop Environment (CDE).

Secure printing

Printers may be assigned different sensitivity labels, thus controlling the range of documents that may be printed on a particular printer. Printing queues management also takes into account sensitivity labels and clearance levels.

Interconnectivity with other systems

Trusted Solaris 8 may communicate with other systems, regardless of whether they support advanced security or not. The settings of the default security level are used when interacting with standard systems or networks not supporting advanced security features.

Trusted CDE

A special, trusted, version of the CDE is available with Trusted Solaris 8. Users will enjoy the same familiar graphical environment, but with additional security measures and features.

Rights profiles

Rights profiles are functionally related procedures that are stored in a special database, which define the authorizations and rights. A default set of rights profiles is supplied with Trusted Solaris, and these may be modified and used in a hierarchical way to fine-tune rights given to a particular user or group of users.

Role-based access control

Role-based access control increases system security through division and restriction of administrative powers, giving only the necessary rights to a particular user or group of users. Additionally, role-based access control introduces the second-level authorization that's performed after the user has passed the first-level OS authentication process.

Discretionary access control

Discretionary access control is used to restrict access to particular resources based on file permissions and access control lists. Unlike standard Solaris, even root is subject to this access control.

Privileges

The privileges system in Trusted Solaris addresses the problem that occurs when a program running as root is exempt from all restrictions. In Trusted Solaris, privileges of software running as a superuser may be controlled in a consistent manner, giving only particular, necessary privileges.

Sensitivity and clearance labels

Sensitivity and clearance labels are the most important concepts in Trusted Solaris. Sensitivity labels are assigned to files, windows, devices, hosts, networks and other objects that may be accessed by users. Clearance labels define the level of trust for a particular user (i.e., the set of objects and actions that particular user may access and perform).

Mandatory access control

Mandatory access control labels permit the division of data and applications into separate compartments depending on the sensitivity of the information. This labeling occurs automatically, which is why it's called mandatory.

Minimum system requirements

To run Trusted Solaris 8, you'll need at least 128 MB of RAM and 1 GB of disk space (2 GB for servers). The HCL (Hardware Compatibility List) is the same for both Solaris 8 and Trusted Solaris 8, so if your current hardware runs Solaris 8, it will accommodate Trusted Solaris 8 without any problems.

Export restrictions

There are no special export restrictions for Trusted Solaris; however, it may not be sold or shipped to countries that are on the U.S. Department of Commerce export restrictions list. Currently, these countries include Afghanistan, Cuba, Iran, Iraq, Libya, North Korea, Serbia/Montenegro (Yugoslavia), Sudan and Syria. Additionally, it's Sun Microsystems' corporate policy not to ship any products to Burma.

Summary

If you're in one of the previously mentioned industries requiring a higher level of security than that provided by the standard Solaris 8 Operating Environment, then Trusted Solaris may be for you. However, although Trusted Solaris is based on standard Solaris, there are some differences in administration and use that take time to master—so be sure your need for more security justifies the increased requirements of Trusted Solaris. ✳

Cross-platform editing with Visual SlickEdit

by Clayton E. Crooks II

Application: Visual SlickEdit
Operating System: Solaris

The use of a high-quality programming editor is a must for anyone who develops for a living. While a multitude of editors are available for UNIX, one product continues to evolve and set itself apart from the competition: Visual SlickEdit. MicroEdge released the first version of the program in the late 1980s, and it since has gone on to win numerous awards and accolades from developers worldwide.

Runs almost everywhere

If you develop on a variety of platforms, you'll appreciate the large number of platforms that SlickEdit supports. It's unparalleled in its platform options with versions available for all Windows variants, OS/2, and a variety of UNIX platforms, including Solaris and Linux. Perhaps even more important than the number of platforms is the fact that the software maintains its look and feel regardless of your OS. This feature dramatically cuts down on user training, as you can learn and master a single interface.

SlickEdit also supports a large variety of development environments, which include an extensive list of popular programming tools such as Visual C++, Visual J++, Visual Café, Delphi, C++ Builder, JBuilder, Watcom C++, Optima++, Visual Age and Visual Basic. You can run and compile any of the languages within SlickEdit and error messages are automatically processed.

Although it is fully extensible and supports most popular languages out of the box, SlickEdit does allow you to add new languages on your own. The following languages are natively supported: C, C++, IDL, Java, JavaScript, HTML, Perl, Pascal/Delphi, Assembly, OS/390 Assembler, COBOL, PL/I, JCL, REXX, DB2, PL/SQL, Transact SQL, Fortran, Ada, dBASE, Modula-2 and AWK. The editor supports the following options with built-in or add-on languages: Syntax Expansion and Indenting, Color Coding, and Dynamic Tagging, to name a few.

Jumpstarting your development

In addition to all the language and OS-specific support, there are a few additional features worth mentioning, one of which is the multiple clipboards that are saved along with your source files. Add to this the intelligent pasting feature that modifies the style and indent level of existing source code to match any section you paste it into, and you have a tremendous combination. SlickEdit also provides unlimited undo and redo and very powerful search and replace functionalities.

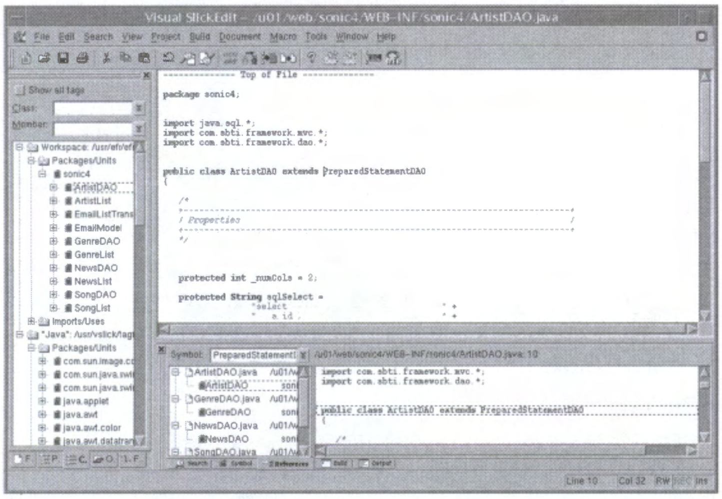
The syntax-highlighting support is easy to configure, and several color schemes are included with the default installation. They provide a good basis for further modification. MicroEdge has included several formatting features, such as the ability to indent a group of selected lines at one time.

With an ever-increasing amount of work moving to the Web, the HTML toolbar is very useful and supports a variety of common functions, such as adding colors, fonts, links and tables. It also includes an FTP client that's particularly useful when you're maintaining or developing Web sites. It can open and save files directly to and from an FTP site so that you'll no longer have to download the file to a local drive before modifying and uploading it, making it seem as if the remote file is locally stored on your computer.

Interface flexibility

The standard SlickEdit interface looks very much like a Windows-based application, as you can see in **Figure A**. The toolbar has options for the majority of the standard functions, such as creating a new file, opening an existing one, saving, printing, and the cut, copy, paste and undo operations that we previously mentioned.

Along with the standard options, the toolbar provides a few advanced features as well. For instance, you can utilize the macro menu to record new macros that are written in a platform-independent language called Slick-C. The platform independence allows you to take macros written on Windows and use them on UNIX or vice versa, a must-have for multi-platform development. You can bind hotkeys to these macros, or call them from the editor command line. It's easy to create your own custom macros, or you can simply alter the existing ones. Another excellent feature is the ability to alter the forms and dialog boxes to your liking.



```
-------------- Top of File --------------

package sonic4;


import java.sql.*;
import com.sbti.framework.mvc.*;
import com.sbti.framework.dao.*;


public class ArtistDAO extends PreparedStatementDAO
{

   /*
   *---------------------------------------------------------------*
   / Properties                                                    /
   *---------------------------------------------------------------*
   */



   protected int _numCols = 2;

   protected String sqlSelect =
                    "select                          " +
                    "   a.id ,                       " +
```

```
import com.sbti.framework.mvc.*;
import com.sbti.framework.dao.*;

public class ArtistDAO extends PreparedStatementDAO
{

   /*
```

Figure A: *The initial user interface in SlickEdit will be comfortable for most GUI users.*

Although the standard interface is well-organized and easy to use, SlickEdit is fully customizable, allowing you to adjust it to your particular needs. The configuration options might be the best feature of the entire package. The primary change you might make is related to the type of editor that SlickEdit should emulate: Visual C++, CUA, Brief, Emacs and vi. If you're familiar with a particular editor, this can be an easy way to get yourself comfortable with SlickEdit.

Conclusion

With all of the customization and features that are present, you might be concerned about the performance that SlickEdit offers. Fortunately, unlike many applications on the market, SlickEdit performs well even on very basic hardware.

If you're a programmer, you'll be hard pressed to find a better editor than SlickEdit—especially if you develop in multiple environments. It's a feature-packed application, and the support offered by MicroEdge is unparalleled. If you're considering SlickEdit, you should check out the Web site at **www.slickedit.com**, where you can obtain a full-featured, time-limited demo version. ✳

DOWNLOAD ftp.elementkjournals.com/sun/mar01

Processing email with aliases

by Don Kuenz

Application: Sendmail 8.11.1
Operating System: Solaris

Sendmail allows you to forward mail or deliver it to mailing lists, files and programs. It's easy to take advantage of this functionality by defining aliases. In this article, we'll show you how to use aliases to process email. We'll begin by explaining how to define aliases. Next we'll tell you how to use an alias to effectively change the delivery address of mail. We'll then cover mailing lists and appending mail to files. Finally, we'll show you how to deliver mail to a program. Our example program allows you to cancel rogue news articles by simply forwarding the article, along with its headers, to a program named newskill.

Defining aliases

You define aliases in a text file named /etc/mail/aliases. **Listing A** shows the contents of our

/etc/mail/aliases file. Lines that start with pound signs (#) contain comments. Sendmail ignores blank lines, which provide white space to enhance readability. The remaining lines contain an alias, followed by a colon, followed by a username, filename or program name.

Although a text file makes it easy for humans to change alias definitions, it also makes it hard for sendmail to efficiently process mail. So, sendmail uses an internal alias database. You use a program named newaliases to compile an aliases text file into an internal database. The following command shows you how the program works:

```
# newaliases -on
/etc/mail/aliases: 36 aliases, longest
➥56 bytes, 744 bytes total
```

The -on argument tells newaliases to validate the right-hand side of each alias. The program also displays whatever errors may be present in your aliases file.

Delivering mail to aliases

The very first alias definition in Listing A tells sendmail to deliver mail addressed to root to a user named don. This ensures that a real person sees all of the mail destined for root.

Listing A: *The contents of our /etc/mail/aliases file*

```
# @(#)aliases 8.2 (Berkeley) 3/5/94
#
#  Aliases in this file will NOT be expanded in
#  the header from Mail, but WILL be visible
#  over networks or from /bin/mail.
#
# >>>>>>>>>>  The program "newaliases"
# >>>>>>>>>>  must be run after
# >> NOTE >>  this file is updated
# >>>>>>>>>>  for any changes to
# >>>>>>>>>>  show through to sendmail.
#
root:                   don

# Basic system aliases--these MUST be present.
MAILER-DAEMON:          postmaster
postmaster:             root

# General redirections for pseudo accounts.
bin:                    root
daemon:                 root
games:                  root
ingres:                 root
nobody:                 root
system:                 root
toor:                   root
uucp:                   root
news:                   root
usenet:                 root

# Well-known aliases.
manager:                root
dumper:                 root
operator:               root

# Trap decode to catch security attacks
decode:                 root

# OFFICIAL CSRG/BUG ADDRESSES

# Ftp maintainer.
ftp:                    ftp-bugs
ftp-bugs:               bigbug@cs.berkeley.edu

# Distribution office.
bsd-dist:               bsd-dist@cs.berkeley.edu

# Fortune maintainer.
fortune:                fortune@cs.berkeley.edu

# Termcap maintainer.
termcap:                termcap@cs.berkeley.edu

# General bug address.
ucb-fixes:              bigbug@cs.berkeley.edu
ucb-fixes-request:      bigbug@cs.berkeley.edu
bugs:                   bugs@cs.berkeley.edu
# END OFFICIAL BUG ADDRESSES

# news killer
newskill:               "|/usr/local/bin/newskill"

# email file
mailfile:               /tmp/mailfile

# mailing lists contained in external files
admin:                  :include:/home/sam/maillist/admin
owner-admin:            sam
programmer:            :include:/home/sam/maillist/programmer
owner-programmer:       sam
biglist:
➥"|/usr/lib/sendmail -oi -odq -first-request samsbiglist"
samsbiglist:            :include:/home/sam/maillist/biglist
biglist-request:        sam
owner-biglist:          sam
owner-owner:            postmaster
```

The next 16 alias definitions deliver all email generated from pseudo users to root, which sendmail in turn forwards to don. Daemons occasionally mail messages to their pseudo user owner, and our aliases make sure that a human sees all messages. The next group of aliases provides default bug reporting. The final group of aliases delivers mail to a program, a file and three mailing lists.

You create an alias that appends mail to a text file by specifying an alias, followed by a colon, followed by a filename. We define an alias named mailfile to append mail to a file named /tmp /mailfile. **Listing B** shows the topmost contents of /tmp/mailfile after we send mail to our mailfile alias. Notice that the first 17 lines contain mail headers. The actual message follows the blank lines that appear after the headers.

The bottom of our aliases file shown in **Listing A** defines three mailing lists. A user named sam maintains all three lists. The first two lists, named admin and programmer, contain local addresses. The `:include:` keyword tells sendmail to obtain the list from three files located under a directory named /home/sam/maillist. This enables user sam to change mail lists without compromising the security of /etc/mail/aliases and using newaliases to constantly generate a new internal database.

The owner-admin, owner-programmer, owner-biglist aliases cause mail bounced from each list to go to user sam. We use the owner-owner alias to designate user postmaster as the default list owner.

The biglist alias defines a list that contains outside mail addresses. Both biglist-request and owner-biglist forward bounced mail to user sam. The biglist and samsbiglist aliases ensure that outside mail sent to biglist bounces to user sam if it encounters errors.

Listing B: *The topmost contents of a file named /tmp/mailfile*

```
From root Sun Dec 24 10:50:41 2000
Return-Path: <don@apollo.crc.wy>
Received: from apollo.crc.wy (eos.crc.wy [192.168.173.145])
   by apollo.crc.wy (8.9.1b+Sun/8.9.1) with ESMTP id KAA02950
   for <mailfile@apollo.crc.wy>;
  ➥Sun, 24 Dec 2000 10:50:41 -0700 (MST)
Message-ID: <3A463770.E186D970@apollo.crc.wy>
Date: Sun, 24 Dec 2000 10:50:40 -0700
From: don <don@apollo.crc.wy>
X-Mailer: Mozilla 4.7 [en] (WinNT; U)
X-Accept-Language: en
MIME-Version: 1.0
To: mailfile@apollo.crc.wy
Subject: [Fwd: REPOST: DNS: Data format error]
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Content-Length: 2886

-------- Original Message --------
From: cmic <michel.marconNOmiSPAM
➥@cetu.equipement.gouv.fr.invalid>
[reposted because of HipClone cancel]
Subject: REPOST: DNS: Data format error
Newsgroups: comp.mail.sendmail
X-Repost-Date: 7 Jul 2000 19:43:58 GMT
Message-ID: <4$-_%%_%_$-$_--%-$@news.noc.cabal.int>
```

Using your mail program to remove news articles

People occasionally post off-topic articles to news groups. You can cancel off-topic articles on an INN news server by invoking `ctlinnd` and using the word `cancel` followed by the `Message-ID` of the article you wish to cancel. Unfortunately, this method becomes quite tedious if you want to cancel more than a few articles. Let's automate the news cancel process by simply forwarding an article to a mail program and making the program perform all of the tedious work.

Our mail program, named newskill, reads its input to obtain the Message-ID of the article that we want to cancel. Newskill expects its input to look like **Listing B**. If you carefully look at the listing, you can see two lines that contain Message-IDs. The first Message-ID appears in the mail header and pertains to the mail itself. The second Message-ID appears in the mail body, after `-------- Original Message --------`, and identifies the article that we wish to cancel.

Newskill reads its input until it sees the second Message-ID, and then calls `ctlinnd cancel <Message-ID>` to remove it. **Listing C** shows the source to newskill.

You can copy the source shown in **Listing C** to a file named newskill.c (or download it from our FTP site), and then compile it with one of the following commands, depending upon the C compiler that you use:

```
cc -o newskill newskill.c
gcc -o newskill newskill.c.
```

You use the following commands to install it:

```
mv newskill /usr/local/bin/newskill
chown news newskill
chmod 4755 newskill
```

By default, sendmail classifies program delivery agents as expensive. An expensive delivery agent causes mail to stay in a queue until an event occurs. During your initial debug phase, you probably want sendmail to immediately deliver mail to programs. This involves changing the default classification.

You can change the default classification by using your favorite editor to edit a file named /etc/mail/sendmail.cf. Now search for the following lines:

```
Mprog, P=/bin/sh, F=lsDFMoqeu9, S=10/30,
➥R=20/40, D=$z:/, T=X-Unix, A=sh -c $u
```

The `e` that appears after `F=` marks this delivery agent as expensive. Simply delete the `e`, save /etc/mail/sendmail.cf, and enter the following commands to force sendmail to use the new configuration:

```
/etc/init.d/sendmail stop
/etc/init.d/sendmail start
```

Now, sendmail immediately delivers mail to programs, instead of queuing it.

One final caveat: To correctly use newskill, you must make sure that the article you're forwarding contains news headers. In Netscape, this involves choosing View | Headers | All from the menu, selecting the article you wish to cancel, and finally choosing Message | Forward from the menu. Address the mail to newskill and send it.

Conclusion

By using aliases, sendmail provides a powerful way to manage delivery from your mail server. You can deliver messages to mailing lists, files and programs. You can also use sendmail to automatically remove news articles from your news server. Hopefully these examples will show you how to take advantage of these features. ✳

Listing C: *A C program named newskill that cancels the second Message-ID that it reads in its input*

```
#include <stdio.h>
#define MAXLINELEN 1024

int main(int argc, char **argv) {
   char buf[MAXLINELEN];
   int bufLen;
   char cmdString[MAXLINELEN + 38];
   char msgIDHeader[] = {"Message-ID: "};
   int msgIDHeaderCount;
   int returnVal;

   msgIDHeaderCount = 0;

   /* loop until error or cancel */
   for (;;) {

      /* read the next line from our standard input */
      if (fgets(buf, sizeof buf, stdin) == NULL) {
         /* return an error if end of file */
         return -1;
      }

      /* does this line contain a Message-ID header? */
      if (strncmp(msgIDHeader, buf, sizeof
      ➥msgIDHeader - 1) == 0) {

         /* if so, increment the header counter */
         msgIDHeaderCount++;

         /* is this the second Message-ID header? */
         if (msgIDHeaderCount > 1) {

            /* if so, call ctlinnd to cancel the article */
            bufLen = strlen(buf);
            if (buf[bufLen - 1] == '\n') {
               buf[bufLen - 1] = 0;
            }
            sprintf(cmdString,
            ➥"/usr/local/news/bin/ctlinnd cancel '%s'",
            &(buf[sizeof msgIDHeader - 1]));
            if (system(cmdString) == 0) {
               return 0;
            } else {
               return 70;
            }
         }
      }
   }
   return 0;
}
```

A better sendmail for Solaris

by Edgar Danielyan

Application: Sendmail 8.11.1
Operating System: Solaris

The sendmail version that comes with the Solaris 8 Operating Environment is outdated, has bugs, and lacks features available in newer releases from the Sendmail Consortium. In this article, we'll show you how to upgrade your Solaris 8 installations to the latest release of sendmail.

Getting the latest release

First, you need to download the latest release of sendmail from ftp.sendmail.org/pub/sendmail. At the time of this writing, the latest release is 8.11.1, which we'll use as an example. To start, let's unpack the distribution with the following command:

```
cat sendmail.8.11.1.tar.gz | gunzip | tar xvf -
```

Then, as a matter of caution, back up your existing sendmail and sendmail.cf with the following command, just in case something goes wrong with the new one:

```
cd /usr/lib
cp sendmail sendmail.sun
cd /etc/mail
cp sendmail.cf sendmail.cf.sun
```

Now, make sure the permissions on certain directories are set as they should be:

```
chmod go-w / /etc /etc/mail /usr
➥/var /var/spool /var/spool/mqueue
chown root / /etc /etc/mail /usr /var
➥/var/spool /var/spool/mqueue
```

If you'll be using Berkeley Database, then obtain, compile and install the latest release from www.sleepycat.com.

Before proceeding, you'll probably want to read sendmail/README to decide what features you want to include in your sendmail installation; include only those you'll be using. Modern sendmail is much easier to build than its ancestor. All you need to do is issue the command `sh Build` in the sendmail-8.11.1 directory. As Solaris is fully supported by sendmail, you shouldn't have any problems compiling it, as long as you have all the necessary libraries and include files in place.

After you've built the binary, you'll probably need to build the configuration file (sendmail.cf). If you do, then use `cd cf/cf` and copy generic-solaris2.mc to config.mc. Run `sh Build config.cf` and copy the resulting config.cf to /etc/mail/sendmail.cf. Edit it as necessary to suit your particular requirements. Assuming you've already stopped your current sendmail, now install the new one with the following commands:

```
cd ../../sendmail
sh Build install
```

You're all set up—at least until the new release comes out! ✳

How much memory does it need?

by George H. Gates

Application: pmap
Operating System: Solaris 2.4 and above

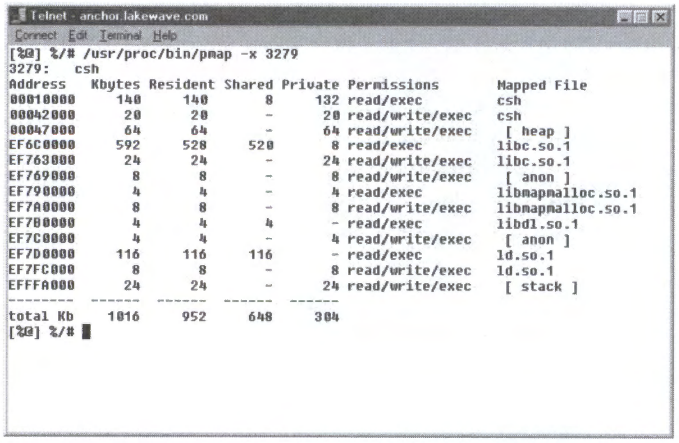
Figuring out how much memory a process needs can be difficult. You have to take into account portions of a process that are shared. This is especially true when you expect to have multiple instances of the same process running. But there's a tool in Solaris that can get the information you need.

Checking process memory with pmap

Pmap is a tool that prints out a map of process memory. You can find it at /usr/proc/bin/pmap. Pmap became available in Solaris 2.4, along with a host of other process information tools. The man page for proc displays all the tools in the /usr/proc/bin directory. The primary way we'll use pmap is with the –x option. This is necessary to break down process memory segments into private and shared resources. **Figure A** shows an example of pmap output.

The memory is broken down by type of usage. The Kbytes column is the virtual memory in use. Resident is the amount of virtual memory actually in physical memory at that time. Resident is then broken down into how much is shared with other processes and how much is private to just that process. All values are displayed in kilobytes.

Memory is also broken down into the type of segment. The first two segments are usually the text (compiled code) and data from the executable itself. The heap is a scratchpad area for the process. The stack is



```
[%@] %/# /usr/proc/bin/pmap -x 3279
3279:   csh
Address   Kbytes Resident Shared Private Permissions       Mapped File
00010000     140      140      8     132 read/exec         csh
00042000      20       20      -      20 read/write/exec   csh
00047000      64       64      -      64 read/write/exec    [ heap ]
EF6C0000     592      528    520       8 read/exec         libc.so.1
EF763000      24       24      -      24 read/write/exec   libc.so.1
EF769000       8        8      -       8 read/write/exec    [ anon ]
EF790000       4        4      -       4 read/exec         libmapmalloc.so.1
EF7A0000       8        8      -       8 read/write/exec   libmapmalloc.so.1
EF7B0000       4        4      4       - read/exec         libdl.so.1
EF7C0000       4        4      -       4 read/write/exec    [ anon ]
EF7D0000     116      116    116       - read/exec         ld.so.1
EF7FC000       8        8      -       8 read/write/exec   ld.so.1
EFFFA000      24       24      -      24 read/write/exec    [ stack ]
--------  ------   ------ ------  ------
total Kb    1016      952    648     304
[%@] %/# █
```

Figure A: *This is a sample of pmap output on the c-shell (csh) process. Only one csh was running.*

About our contributors

Clayton E. Crooks II is a self-employed computer consultant living in Knoxville, Tenn. He's married with one child. His hobbies include game development, 3-D modeling and any athletic activity he can find time for.

Edgar Danielyan is currently self-employed. His qualifications include Cisco Certified Network Associate, diploma in company law from the British Institute of Legal Executives, and certified paralegal from the University of Southern Colorado. He's been working as a network administrator and manager of a top-level domain of Armenia. 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, USENIX, CIPS, ISOC and IPG, to name a few. He can be reached at **edd@danielyan.com**.

George H. Gates has been a contract instructor for Sun Microsystems for the past three years. He teaches a range of classes including System Administration level I & II, Network Administration, Enterprise System Performance Management and Enterprise 10000 System Administration. You can reach him at **ggates03@twcny.rr.com**.

Don Kuenz works at Computing Resources Company (**gtcs.com/crc**). They provide programming, administration and hardware for Sun and PC platforms. You can reach Don at **kuenz@gtcs.com**.

Alan Orndorff has been working with computers since 1990. He's using Solaris as a platform for Lotus Notes and at home in his spare time. He currently lives in San Francisco and can be reached at **dwarf@solarisresources.com**.

Inside Solaris™
Tips & Techniques for users of Sun Solaris

Inside Solaris (ISSN 1081-3314) is published monthly by Element K Journals, a division of Element K Press, 500 Canal View Boulevard, Rochester, N.Y., 14624.

Customer Relations

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

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

Element K Journals Customer Relations
500 Canal View Boulevard
Rochester, NY 14623

Or contact Customer Relations via Internet email at journals@element-k.com.

Editorial

Editor	Garrett Suhm
Assistant Editor	Jill Suhm
Editorial Director	Michelle Rogers
Managing Editor	Craig Watkins
Copy Editors	Rachel Krayer
	Glenna Lechner
Contributing Editors	Clayton E. Crooks II
	Edgar Danielyan
	George H. Gates
	Don Kuenz
	Alan Orndorff
Graphic Designer	Rachel J. King
Cover and Content Design	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) 272-0064

Or contact us via Internet email at inside_solaris@elementkjournals.com.

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

Element K Journals

General Manager	Kelly Baptiste
Manager of Customer Relations	Nicole Pate
Manager of Operations	Cristal Haygood
Manager of Graphic Design	Ian Caspersson
Manager of Product Marketing	Brian Cardona
Product Marketing Manager	Robin Mullins

Postmaster

Periodicals postage paid in Rochester, N.Y., and additional mailing offices.

Postmaster: Send address changes to

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

Price

Domestic	$129/yr ($11.00 each)
Outside U.S.	$149/yr ($13.00 each)

Our Canadian GST# is: R140496720. CPM# is: 1446703. QST# is: 1018491237.

Back Issues

To order a back issue from the last six months, call Customer Relations at (800) 223-8720. Back issues cost $11.00 each, $13.00 outside the U.S. You can pay with MasterCard, VISA, Discover or American Express.

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 journals@element-k.com.

Coming up...

- Citrix WinFrame and Solaris
- Using the stow utility

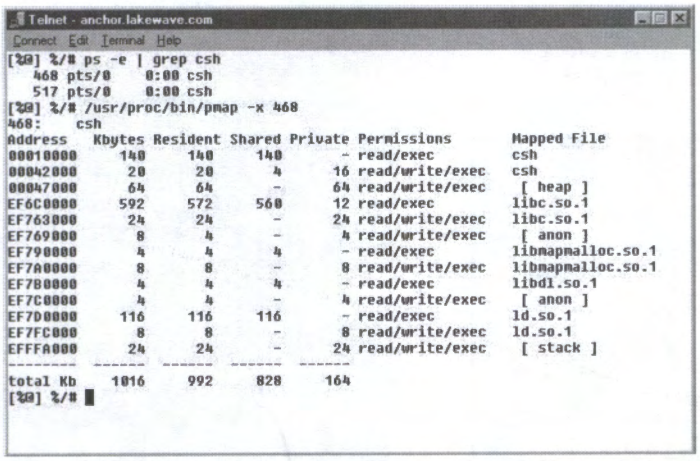
PERIODICALS MAIL

2096

local storage for the executing function. Segments that contain .so. are shared libraries. There may also be other types of segments present. The text, portions of the data and shared libraries normally have shared portions. Stack and heap are private to that process. All the segments are included in the total, a summary of the entire process. We'll use the total in memory sizing calculations.

Multiple instances of the same process

One thing to keep in mind is the effect of having multiple instances of the same process running.



```
[%@] %/# ps -e | grep csh
   468 pts/0    0:00 csh
   517 pts/0    0:00 csh
[%@] %/# /usr/proc/bin/pmap -x 468
468:    csh
Address   Kbytes Resident Shared Private Permissions       Mapped File
00010000     140      140    140       - read/exec         csh
00042000      20       20      4      16 read/write/exec   csh
00047000      64       64      -      64 read/write/exec    [ heap ]
EF6C0000     592      572    560      12 read/exec         libc.so.1
EF763000      24       24      -      24 read/write/exec   libc.so.1
EF769000       8        4      -       4 read/write/exec    [ anon ]
EF790000       4        4      4       - read/exec         libmapmalloc.so.1
EF7A0000       8        8      -       8 read/write/exec   libmapmalloc.so.1
EF7B0000       4        4      4       - read/exec         libdl.so.1
EF7C0000       4        4      -       4 read/write/exec    [ anon ]
EF7D0000     116      116    116       - read/exec         ld.so.1
EF7FC000       8        8      -       8 read/write/exec   ld.so.1
EFFFA000      24       24      -      24 read/write/exec    [ stack ]
--------  ------   ------ ------  ------
total Kb    1016      992    828     164
[%@] %/# █
```

Figure B: *This is a sample of pmap output on the c-shell (csh) process. Two csh processes were running.*

Let's assume we're using c-shell as the main user login shell. We'll use the example of figuring out how much memory is needed for the c-shell process based on user count. While **Figure A** shows pmap output with only one csh process running, **Figure B** shows pmap output with two c-shell processes running.

You'll notice that the shared and private portions have changed between **Figure A** and **Figure B**. Even though we have two c-shells in execution, there need be only one copy of the text in memory, as well as the shared libraries they're using.

Putting it all together

We can now figure out the amount of memory necessary based on the user count. Making sure we have more than one instance of the process running, we can take the total private memory x user count + total shared memory. For example, using the totals from **Figure B**, if we expect 20 users logged on at once, each with a c-shell running, we'd need 20 x 164 + 828. The total is 4,108 KB. It would be easy to update memory needs based on expanding user load.

Conclusion

Understanding process memory usage can be important for determining physical memory needs. Using pmap can help you figure it out based on not only current usage, but expected usage. ✳

Inside Solaris survey winners

Congratulations to our winners! Three readers were randomly selected from all recently returned *Inside Solaris* reader surveys. They are:

- Joseph Furmanske of Pittsburgh, Pa.
- Robert Kowalke of Chesapeake, Va.
- Peter Skibar of Edison, N. J.

Each will receive a complimentary one-year extension to their current *Inside Solaris* subscription. These surveys are important tools in our quest to provide a quality publication. Thanks to everyone who took the time to fill one out.