Chapter 5

Configuring User Accounts and Managing User Issues

User Login Administration

Creating and deleting user accounts are two of the most common system administration tasks you will perform. You should read and understand this section before you undertake to manipulate user accounts. The graphical System Manager tool (available on graphics workstations only) is the most convenient tool for these tasks. The System Manager is described in the *Personal System Administration Guide*.

User ID Numbers

Each user account has a user ID number. These numbers are unique on each workstation and should be unique throughout your entire site. A user ID number for an account is kept in the third field of the */etc/passwd* file.

After you close a user account, do not reuse that account's user ID number. It is possible that files somewhere on a system could still be owned by the ID number, or files may be placed on the system from an old backup tape. These sorts of file would be compromised by associating a new user with an old ID number. In general, the rule is that the ID number is the permanent property of the user to whom it is assigned. For more information on user ID numbers, see the *passwd*(4) reference page.

Group ID Numbers

Each user account belongs to a group of users on the system. Users with similar interests or jobs can belong to the same group. For example, members of the publications department might belong to group *pub*. The benefit to this arrangement is that it allows groups of related users to share files and resources without sharing those files or resources with the entire system community.

Each group has a group ID number. These numbers are unique on each system and should be unique throughout the entire site. Like user IDs, you should not reuse group IDs.

When you create a file, it is assigned your group ID. You can change the group ID of a file with the *chgrp*(1) command. By manipulating the permissions field of the file, the owner (or someone with the effective user-ID of the owner) can grant read, write, or execute privileges to other group members.

Information about groups is kept in the */etc/group* file. A sample entry from this file is shown and explained below:

```
raccoons::101:norton,ralph
```

Each entry is one line; each line has the following fields:

group name — The group name can be any length, though some commands will truncate the name to 8 characters. The first character must be alphabetic.

password — The password field may contain an encrypted password. An empty field, as in the above example, indicates that no password is required. The *passwd*(1M) command cannot be used to create or modify a group password. To place a password on a group, you must use the *passwd* command to encrypt a password. (Use a test user account created specifically for this purpose and then delete the test account.) Then, copy that encrypted password verbatim from the */etc/passwd* file into the */etc/group* entry you wish to protect with the password. Users specifically listed as group members in the */etc/group* file entry will not be required to give the password, but other users will be so required when they attempt to change groups to the protected group with the *newgrp*(1) command. Password protection, though, is rarely used on user groups.

group ID — The group ID is a number from 0 to 60,000. The number must not include a comma. Numbers below 100 are reserved for system accounts.

login names — The login names of group members are in a comma-separated list.

For complete information on user groups, see the *group*(4) reference page.

Adding User Accounts Using Shell Commands

Occasionally, you may have to add a user account manually; in other words, without using the automated tools such as the System Manager. This method is the default for server administrators, but all administrators should understand the process in case a problem develops with some part of the automated tools or if you wish to design your own scripts and programs for administering user accounts at your site. Be sure to check your work with the *pwck*(1M) command.

Follow these steps to add a user manually:

1. Log in as **root**.
2. Edit the file */etc/passwd* with your preferred text editor.

   The file */etc/passwd* has one line for each account on the system. Each line contains seven fields, and each field is separated by a colon. The lines look similar to this:

   ```
   ralph:+:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```
3. Copy one of the lines (for example, the last line in the file) and add it to the end of the file.
4. Change the first field (*ralph* in this example) to the name of the new account; for example, *alice*.
5. Remove any characters between the first colon after the account name and the second colon. Deleting these characters removes the password from an account. Either you or the new user can add a password later.
6. The next field (in this case "103") is the user ID of the new account. Change it to a number 1 greater than the current highest user ID on your system. You should not use user ID numbers between 0 and 100, as these are reserved for system use.
7. The next field (in this case "101") is the group ID number of the new account. Check the file */etc/group* and pick a suitable group ID for the new user account. The */etc/group* file lists all the groups on the system by group ID, followed by a list of the current users who belong to that group.
8. Change the next field (in this case *Ralph Cramden*) to the name of the new user, in this case *Alice Cramden*. If you wish, you can add an "office" and "phone number" to this field. After the user's name, add a comma, then the office location, another comma, and the phone number. For example:

   ```
   :Alice Cramden, Brooklyn, (212) 555-1212:
   ```

Actually, you can put any information you wish in these fields. The fields are interpreted by the *finger*(1) program as "user name, office, phone number."

9. The next field (in this case */usr/people/ralph*) is the location of the user's home directory. Change this field to reflect the name of the new user's account. In this example, you would change */usr/people/ralph* to */usr/people/alice*.

10. The last field (in this example */bin/csh*) is the user's login shell. For most users, the C shell (*/bin/csh*), Korn Shell (*/bin/ksh*), or Bourne shell (*/bin/sh*) is appropriate. You should leave this field unchanged, unless you wish to use a different or special shell. Special shells are discussed in "Configuring Special Login Shells" on page 101. Once you have selected a shell, you are finished editing */etc/passwd*.

11. Write the changes you made and exit the file.

    The next step, which is optional, is to add the new user to the file */etc/group*. A user can be a member of a group without being listed in the */etc/group* file.

12. If you want to maintain a list of the groups to which users belong, edit the file */etc/group*. You should see some lines similar to this:

```
sys::0:root,bin,sys,adm
root::0:root
daemon::1:root,daemon
bin::2:root,bin,daemon
adm::3:root,adm,daemon
mail::4:root
uucp::5:uucp
rje::8:rje,shqer
lp:*:9:
nuucp::10:nuucp
bowling:*:101:ralph
other:*:102:
```

    Place the name of the new account (in this example *alice*) after any of the groups. Separate the account name from any other account names with a comma, but not with blank spaces. For example:

```
bowling:*:101:ralph,alice
```

    Adding account names to the */etc/group* file is optional, but it is a good way to keep track of who belongs to the various system groups.

    Also, you can assign an account to more than one group by placing the account name after the names of the various groups in */etc/group*. The user can change group affiliations with the *newgrp*(1) and *multgrps*(1) commands.

13. When you finish editing */etc/group*, write your changes and exit the file.

    The next step is to create the new user's home directory and copy shell startup files over to that directory.

14. Use the *mkdir*(1) command to create the user's home directory. For example, to create a home directory for the user "alice":

    mkdir /usr/people/alice

    Make the directory owned by user *alice*, who is in group *bowling*:

    chown alice /usr/people/alice

    chgrp bowling /usr/people/alice

    Make sure the new home directory has the appropriate access permissions for your site. For a site with relaxed security:

    chmod 755 /usr/people/alice

    For more information on the *chown*(1), *chgrp*(1), and *chmod*(1) commands, see the respective reference pages.

15. Copy the shell startup files to the new user's home directory.

    If the new account uses the C shell:

    cp /etc/stdcshrc /usr/people/alice/.cshrc

    cp /etc/stdlogin /usr/people/alice/.login

    If the new account uses the Korn or Bourne shell:

    cp /etc/stdprofile /usr/people/alice/.profile

16. You can make these shell startup files owned by the user, or leave them owned by *root*. Neither approach affects how the user logs in to the system, although if the files are owned by *root*, the user is less likely to alter them accidentally and be unable to log in.

    To give a user complete access to his or her shell startup files, use the *chmod* command. For C shell:

    chmod 755 /usr/people/alice/.cshrc /usr/people/alice/.login

    For Korn or Bourne shell:

    chmod 755 /usr/people/alice/.profile

    Remember to check for any other user files that may be owned by *root* in the user's directory, and change those, too.

17. Issue the *pwck*(1M) command to check your work. This command performs a simple check of the */etc/passwd* file and makes sure that no user ID numbers have been reused and that all fields have reasonable entries. If your work has been done correctly, you should see output similar to the following:

```
sysadm:*:0:0:System V Administration:/usr/admin:/bin/sh
        Login directory not found
auditor::11:0:Audit Activity Owner:/auditor:/bin/sh
        Login directory not found
dbadmin::12:0:Security Database Owner:/dbadmin:/bin/sh
        Login directory not found
tour::995:997:IRIS Space Tour:/usr/people/tour:/bin/csh
        Login directory not found
4Dgifts::999:998:4Dgifts Acct:/usr/people/4Dgifts:/bin/csh
        First char in logname not lower case alpha
        1 Bad character(s) in logname
        Login directory not found
nobody:*:-2:-2::/dev/null:/dev/null
        Invalid UID
        Invalid GID
```

These messages are normal and expected from *pwck*. All errors generated by *pwck* are described in detail in the *pwck*(1M) reference page.

Adding User Groups Using Shell Commands

Follow these steps to add a group to the system manually:

1. Log in as **root**.
2. Edit the file */etc/group*. The file contains a list of groups on the system, one group per line. Each line contains the name of the group, an optional password, the group ID number, and the user accounts who belong to that group.

   For example, to create a group called *raccoons*, with a group ID of 103, place this line at the end of the file:

   `raccoons:*:103:`
3. If there are users who should belong to the group, add their names in the last field. Each name should be separated by a comma, for example:

   `raccoons:*:103:ralph,norton`
4. Write and exit the file. Make sure the group IDs in the file */etc/passwd* file match those in the */etc/group* file.

For more information on user groups, see the *group*(4) reference page.

Changing a User's Group

To change a user's group affiliation, perform these steps:

1. Login as **root**.
2. Edit the file */etc/group*. Place the user's account name on the line corresponding to the desired group. If the account name appears as a member of another group, remove that reference unless you want the account to be a member of both groups.
3. Write and exit the file */etc/group*.
4. Edit the file */etc/passwd*.
5. Find the user's entry in the file.
6. Change the old group ID on that line to the new group ID. The group ID is the fourth field (after the account name, password, and user ID).
7. Write and exit the file.

The user's group affiliation is now changed. Remind the user to change the group ownership on his or her files. If you prefer, you can perform this task yourself as **root** using the *find*(1) and *chgrp*(1) commands. See "Using find to Locate Files" on page 21 for more information.

Deleting a User from the System

This procedure deletes the user's home directory and all the files in and below that directory. If you only wish to close or disable a user account but preserve the user's files and other information, see "Locking a User Account" on page 86.

To delete a user's account completely, follow these steps:

1. Log in as **root**.
2. If you think you might need a copy of the user's files later on, make a backup copy of the directory (for example, on cartridge tape using *tar*(1) or *cpio*(1)).

3. Edit the */etc/passwd* file and replace the encrypted password (or "+" sign if you are using shadow passwords) with the following string:

   ```
   *ACCOUNT CLOSED*
   ```

   It is imperative that the asterisks shown in this example be used as shown. An asterisk in the encrypted password field disallows all logins on that account. Alternately, you can lock an account by using fewer than thirteen characters in the password field, but it is better to use the asterisks and an identifiable lock message.

4. Use *find*(1) to locate all files on the system that are owned by the user and remove them or change their ownership. Information on the use of *find*(1) is provided in "Using find to Locate Files" on page 21 and in the *find*(1) reference page.

Deleting a Group from the System

Deleting a group from the system is done in the following steps:

1. Edit the */etc/group* file and change the desired entry to a new but unused name, for example, you might change the group *bigproject* to *bigproject.closed*.
2. Edit the */etc/passwd* file and remove the group from the user entries wherever it exists.
3. Use *find*(1) to find all files and directories with the old group affiliation and change the affiliation to a group that is still in use with the *chgrp*(1) command.

Locking a User Account

If you wish, you can close a user's account so that nobody can log in to it or *su* to that user's ID number.

If you expect that the user will again require access to the system in the near future, close an account in the manner described below rather than removing it from the system completely (as described in "Deleting a User from the System" on page 85).

To close an account, follow these steps:

1. Log in as **root**.
2. Edit the file */etc/passwd*. Find the user's account entry.

3. Make the entry a comment by placing a number sign at the beginning of the line. For example:

   ```
   # ralph:+:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

4. As an added measure of security, you can replace the encrypted password (the second field in the entry) with a string that cannot be interpreted as a valid password. For example:

   ```
   # ralph:*:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

   This has the added benefit of reminding you that you deliberately closed the account.

5. If necessary, you can also close off the user's home directory with the following commands:

   chown root /usr/people/ralph

   chgrp bin /usr/people/ralph

   chmod 700 /usr/people/ralph

The user's account is now locked, and only *root* has access to the user's home account.

```
cramden:x:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
```

Temporarily Changing User Groups

Normally users are members of only one group at a time. However, users can change groups using the *newgrp*(1) command. Changing groups is sometimes useful for performing administrative tasks.

The superuser can belong to any group listed in the */etc/groups* file. Other users must be listed as members of a group in order to temporarily change groups with *newgrp*.

You can belong to multiple groups simultaneously by invoking the *multgrps*(1) command. In this case, files you create have their group IDs set to the group you were in before you issued the *multgrps* command. You have group access permissions to any file whose group ID matches any of the groups you are in.

You can only change groups to a group you are affiliated with in the */etc/groups* file. To determine which groups you belong to, use the *id*(1) command.

Changing User Information

This section covers the following procedures:

- "Changing a User's Login Name" on page 88
- "Changing a User's Password" on page 89
- "Changing a User's Login ID Number" on page 90
- "Changing a User's Default Group" on page 90
- "Changing a User's Comments Field" on page 91
- "Changing a User's Default Home Directory" on page 91
- "Changing a User's Default Shell" on page 92

This section tells you how to change the values for an individual user's login information. You cannot use these commands for a login you installed as an NIS-type entry. If the login account is an NIS type, you must change the master password file on the network server. See the chapter "The Network Information Service (NIS)" in the *NFS User's Guide* for more information about NIS.

Changing a User's Login Name

To change a user's login name, perform the following steps:

1. Edit the */etc/passwd* file and change the entry for the user's login to reflect the new login name. For example, to change the login name *cramden* to *ralph*, find the line:

   ```
   cramden:x:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
   ```

   Change the name field and the default directory field as follows:

   ```
   ralph:x:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

   For consistency's sake, the home directory should always have the same name as the user account. If your system has shadow passwords enabled, you will see the character "x" in place of the encoded password in the user's entry. For more information on shadow passwords, see the *IRIX Admin: Backup, Security, and Accounting* guide.

   When you save and exit the file, you may see an error message that the file is "read-only" or that write permission is denied. This is a protection that IRIX puts on the */etc/passwd* file. Use the command:

   ```
   :w!
   ```

in the *vi*(1) editor to override this protection. If you are using *jot*(1), the file can be saved with no difficulty. For complete information on the commands available in *vi*(1), see the *vi*(1) reference page.

2. If your system has shadow passwords enabled, edit the */etc/shadow* file next. Look for the line that begins with the user name you wish to change. In this example, the line would look like this:

   `cramden:XmlGDVKQYet5c::::::`

   Change the name in the entry to "ralph" as follows:

   `ralph``:XmlGDVKQYet5c::::::`

   When you have made the change, save and exit the file. As with */etc/passwd*, if you are using *vi*(1), you may encounter an error message.

3. Go to the directory that contains the user's home directory and change the name of the home directory to match the user's new login name. Use the command:

   `mv cramden ralph`

4. Since IRIX identifies the files owned by the user "cramden" by the user ID number, rather than by the login name, there should be no need to change the ownership.

Changing a User's Password

Occasionally, a user forgets his or her password. To solve the problem, you must assign that user a temporary password, then have the user change the temporary password to something else. This is because there is no easy way to guess a forgotten password.

To assign a new password, perform these steps:

1. Log in as **root**.

2. Use the *passwd* command to change the password for the user's account.

   For example, if the user *ralph* forgets his password, enter:

   `passwd ralph`

3. Follow the screen prompts: `(`*`The password entered is not echoed.`*`)`

   `New password: `**`2themoon`**

   `Re-enter new password: `**`2themoon`**

   Because you are logged in as the superuser (*root*), you are not prompted for an old password.

The user's password is now changed to "2themoon." The user should immediately change the password to something else.

Changing a User's Login ID Number

It is not recommended to change user login ID numbers. These numbers are crucial in maintaining ownership information and responsibility for files and processes. However, if for some reason you must change a user's login ID number, perform the following steps:

1. Make a complete backup tape of the user's home directory and any working directories he or she may have on the system.
2. Lock the user's account by placing a number sign (#) at the beginning of the line, and an asterisk (*) in the password field of the */etc/passwd* file. Do not delete the entry, as you will want to keep it as a record that the old user ID number was used and should not be reused. When you are finished, the entry should look like this:

   ```
   # ralph:*:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```
3. Completely remove the user's home directory, all subdirectories, and any working directories the user may have.
4. Use the following command from your system's root directory to find any other files the user may own on the system and display the filenames on the console:

   ```
   find . -user name -print
   ```

   Archive and remove any files that are found.
5. Create a new user account using the instructions provided in this chapter. It may have the same name as the old account, but it is better to change names at the same time, to avoid confusion. Use the new user ID number.
6. Restore the home directory, working directories, and any other files you located from the backup you made. If necessary, use the *chown*(1) and *chgrp*(1) commands to set the new user ID correctly.

Changing a User's Default Group

A user can be a member of many different groups on your system, but only one group is the default group. This is the group that the user begins with at login time. To change the default group at login time, simply edit the */etc/passwd* file and find the appropriate line. For example:

```
cramden:+:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
```

To change the default group from 101 to 105, simply change the field in the passwd file entry as follows:

```
cramden:+:103:105:Ralph Cramden:/usr/people/cramden:/bin/csh
```

Be certain before you make any change, however, that group 105 is a valid group in your */etc/groups* file and that the user is a member of the group. For more information on creating groups, see "Adding User Groups Using Shell Commands" on page 84.

Changing a User's Comments Field

There is a field in each entry in */etc/passwd* for comments about the user account. This field typically contains the user's name and possibly his or her telephone number or desk location. To change this information, simply edit the */etc/passwd* file and change the information. Note only that you cannot use colon (:) within the comments, since IRIX will interpret these as ending the comments field. For example, consider this entry:

```
cramden:x:103:101:Ralph Crumdin:/usr/people/cramden:/bin/csh
```

It would not be long before Ralph came to the administrator and requested to have the misspelling of his name corrected. In this case, you would change the line to read:

```
cramden:x:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
```

Changing a User's Default Home Directory

The next field in an */etc/passwd* entry specifies the user's home directory. This is the directory that the user is placed in at login time, and the directory that is entered in the shell variable *$HOME*. For complete information on shell variables, see the reference pages for the shell you are using (typically *csh*(1), *sh*(1), *tcsh*(1), or *ksh*(1)). Home directories are typically placed in */usr/people*, but there is no reason why you cannot select another directory. Some administrators select a different directory to preserve file system space on the */usr* file system, or simply because the users have a strong preference for another directory name. In any case, the procedure for changing the home directory of a user is quite simple. Follow these steps:

1. Log in as **root**.
2. Edit the */etc/passwd* file and look for the user entry you wish to change. For example:

   ```
   cramden:x:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
   ```

In this example, the home directory is */usr/people/cramden*. If you wish to change the home directory to a recently added file system called *disk2*, change the entry to read:

```
cramden:x:103:101:Ralph Cramden:/disk2/cramden:/bin/csh
```

When you have made your changes, write and exit the file.

3. Create the directory in the new file system and move all of the files and subdirectories to their new locations. When this is done, remove the old home directory and the process is finished.
4. Be sure that you notify your users well in advance if you plan to change their home directories. Most users have personal aliases and programs that may depend on the location of their home directory. As with all major changes to your system's layout, changing home directories should not be done for trivial reasons, as it can cause serious inconvenience to your users.

Changing a User's Default Shell

To change the default shell, follow these steps:

1. Log in as **root**.
2. Edit the */etc/passwd* file and change the field that names the user's default shell. For example, in the *passwd* entry:

   ```
   ralph:x:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

   The default shell is */bin/csh*. To change the user's default shell, change the field in the passwd entry to the desired program. For example, to change Ralph's shell to */bin/sh*, edit the line to look like this:

   ```
   ralph:x:103:101:Ralph Cramden:/usr/bin/ralph:/bin/sh
   ```

3. Save and exit the */etc/passwd* file. When the user next logs in, the new default shell will be used.
4. Note that you can use any executable program as the default shell. IRIX simply executes the given program when a user logs in. Note also that using a program other than a command shell such as *csh* or *sh* can cause problems for the user. When the program identified as the default shell in the *passwd* file exits, the user is logged out. Therefore, if you use */bin/mail* as the default shell, when the user exits *mail*, he or she will be logged out and will not be able to perform any other work.

Configuring The User's Environment

A user's environment is determined by certain shell startup files. For C shell users, these are the files */etc/cshrc* and, in their home directories, *.cshrc* and *.login*. For Korn Shell users, these are the */etc/profile* file and the *.profile* file in their home directories. For Bourne shell users, these are the files */etc/profile* and, in their home directories, *.profile*.

Shell startup files configure a user's login environment and control aspects of sub-shells created during a login session.

Available Login Shells

The following login shells are provided with IRIX:

/bin/csh — This shell provides a history mechanism (that is, it remembers commands you enter); job control (you can stop processes, place them in background, and return them to foreground); and the ability to use wildcard characters to specify file names. Users can construct sophisticated commands and scripts using a C programming language-like syntax. For a complete description of this shell, see the *csh*(1) reference page.

/bin/ksh — This shell is an expansion of the best features of the Bourne shell and the C shell, and allows for command line editing, job control, programming from the shell prompt in the shell language, and other features. For a complete description of this shell, see the *ksh*(1) reference page.

/bin/sh — This is a simpler shell than *csh* and is also called the *Bourne* shell after its principle designer. Bourne shell does not contain any kind of history mechanism and uses a somewhat different syntax than *csh*. It does make use of wildcard characters and is smaller and faster to invoke than *csh*. For a complete description of this shell, see the *sh*(1) reference page.

/bin/rsh — This is a restricted shell, which limits the commands a user can type. The *rsh* command syntax is identical to *sh*, except that users cannot perform the following:

- change directories
- use the *ls*(1) command
- set the shell search path ($PATH)

- specify path or command names containing /
- redirect output (> and >>)

The restrictions against */bin/rsh* are enforced after *profile* has been executed.

For complete information about these shells, see the *ksh*(1), *csh*(1), and *sh*(1) reference pages. The *rsh* restricted shell is described on the *sh*(1) reference page.

Note: Two shells called *rsh* are shipped with IRIX. */bin/rsh* is the restricted shell. The other shell, in */usr/bsd/rsh*, is the Berkeley remote shell. Be careful not to confuse the two.

The various startup files that configure these shells are described in the next sections.

C Shell Configuration Files

When a C shell user logs in to the system, three startup files are executed in the following order:

1. The */etc/cshrc* file.

   This is an ASCII text file that contains commands and shell procedures, and sets environment variables that are appropriate for all users on the system. This file is executed by the *login* process.

   A sample */etc/cshrc* is shown below:

   ```
   # default settings for all users
   # loaded <<before>> $HOME/.cshrc
   umask 022
   if ($?prompt) cat /etc/motd
   set mail=$MAIL
   if ( { /bin/mail -e } ) then
    echo 'You have mail.'
   endif
   ```

   In the example, several commands are executed:

   - the message of the day is displayed if a prompt exists
   - the location of the user's mail file is set
   - a message informs the user if he or she has mail

2. The individual user's *.cshrc*.

   This file is similar to */etc/cshrc*, but is kept in the user's home directory. The *.cshrc* file can contain additional commands and variables that further customize a user's environment. For example, use this file to set the shell prompt for the user. The *.cshrc* file is executed whenever a user spawns a subshell. A sample *.cshrc* is shown below:

   ```
   set prompt  = "Get back to work: "
   set filec
   set history = 20
   ```

   In this example, the user's prompt is set, automatic filename completion is turned on, and the length of the history of recently issued commands is set to 20.

3. The *.login* file.

   This is an executable command file that resides in the user's home directory. The *.login* also customizes the user's environment, but is only executed once, at login time. For this reason, you should use this file to set environment variables and to run shell script programs that need be done only once per login session. A sample *.login* is shown below:

   ```
   eval `tset -s -Q`
   umask 022
   stty line 1 erase '^H' kill '^U' intr '^C' echoe
   setenv DISPLAY wheeler:0
   setenv SHELL csh
   setenv VISUAL /usr/bin/vi
   setenv EDITOR /usr/bin/emacs
   setenv ROOT /
   set path = (. ~/bin /usr/bsd /bin /usr/bin /usr/sbin \ /usr/bin/X11
   /usr/demos /usr/local/bin)
   ```

   In this example, the user's terminal is further initialized with *tset*(1), then the file creation mask is set to 022. Some useful key bindings are set, using the command *stty*(1). The user's default display is set to be on the console screen of the machine called "wheeler." Several important environment variables are set for commonly used utilities and the file system point of reference. Finally, the default path is expanded to include the user's own binary directory and other system directories.

For information on the shell programming commands used in these examples, see the *csh*(1) reference page.

Bourne and Korn Shell Configuration Files

When a Bourne or Korn shell user logs in to the system, two startup files are executed in the following order:

1. The */etc/profile* file.

   This is an ASCII text file that contains commands and shell procedures and sets environment variables that are appropriate for all users on the system. This file is executed by the *login* process.

   A sample */etc/profile* is shown below:

```
# Ignore keyboard interrupts.
trap "" 2 3
# Set the umask so that newly created files and directories will be
readable
# by others, but writable only by the user.
umask 022
case "$0" in
*su )
# Special processing for ``su -'' could go here.
;;
-* )
# This is a first time login.
#
# Allow the user to break the Message-Of-The-Day only.
trap "trap '' 2" 2
cat -s /etc/motd
trap "" 2
# Check for mail.
if /bin/mail -e
then
echo "you have mail"
fi
;;
esac
trap 2 3
```

   In the example, several commands are executed:

   - keyboard interrupts are trapped
   - the user's *umask* is set to 022—full permission for the user, read and execute permission for members of the user's group and others on the system

- if the user is logging in for the first time, the message of the day (*/etc/motd*) is displayed, and the user is notified if he or she has mail

2. The individual user's *.profile*.

   This file is similar to */etc/profile*, but is kept in the user's home directory. The *.profile* file can contain additional commands and variables that further customize a user's environment. It is executed whenever a user spawns a subshell.

   A sample *.profile* is shown below:

```
# Set the interrupt character to <Ctrl-C> and do clean backspacing.
stty intr '' echoe
# Set the TERM environment variable
eval `tset -s -Q`
# List files in columns if standard out is a terminal.
ls() { if [ -t ]; then /bin/ls -C $*; else /bin/ls $*; fi }
PATH=:/bin:/usr/bin:/usr/sbin:/usr/bsd:$HOME/bin:.
EDITOR=/usr/bin/vi
PS1="IRIX> "
export EDITOR PATH PS1
```

   In this example:

   - the interrupt character is set to **<Ctrl-C>**
   - the TERM environment variable is set with *tset*(1)
   - a function called *ls* is defined so that when the user enters *ls* to list files in a directory, and the command is issued from a terminal or window, the *ls* command is invoked with the **-C** option
   - the environment variables PATH and EDITOR are set
   - the user's prompt (PS1) is set to *IRIX***>**

For information on the shell programming commands used in these examples, see the *ksh*(1) and *sh*(1)reference pages.

Configurable Shell Environment Variables

Every shell uses a series of variables that hold information about the shell and about the login account from which it originated. These variables provide information to other processes as well as to the shell itself.

Collectively, these environment variables make up what is called the shell’s *environment*. The basic concepts of environment variables and an environment are the same for all types of IRIX shells, although the exact method of creating and manipulating the environment variables differs.

A basic set of environment variables includes where in the IRIX file system to search for commands (PATH), the location of the home directory of the user’s account (HOME), the present working directory (PWD), the name of the *terminfo* description used to communicate with the user’s display screen or terminal (TERM), and some other variables.

When a process (shell) begins, the *exec*(2) system call passes it an array of strings, called the *environment*.

Since *login* is a process, the array of environment strings is made available to it. To look at your current shell environment, use the *printenv* command. A typical C shell environment might look something like this:

```
LOGNAME=trixie
PWD=/usr/people/trixie
HOME=/usr/people/trixie
PATH=.:/usr/people/trixie/bin:/usr/bsd:/bin:/etc:/usr/sbin:
/usr/bin: /usr/local/bin:
SHELL=/bin/csh
MAIL=/var/mail/trixie
TERM=iris-ansi
PAGER=more
TZ=EST5EDT
EDITOR=emacs
DISPLAY=myhost:0
VISUAL=vi
```

For C shell users, these variables are set in either the */etc/cshrc*, *.cshrc*, or *.login* startup files. For Korn and Bourne shell users, these variables are set in either the */etc/profile* or *.profile* startup files.

The default environment variables that are assigned for C shell users, if no others are set in any of the startup files, are:

- HOME
- PATH
- LOGNAME

- SHELL
- MAIL
- TZ
- USER
- TERM

Other processes use this information. For example, user *trixie*’s terminal is defined as an *iris-ansi* (TERM=iris-ansi). When the user invokes the default visual editor *vi*(1), *vi* checks this environment variable, then looks up the characteristics of an *iris-ansi* terminal.

New variables can be defined and the values of existing variables can be changed at any time with the *setenv* command (C shell only). For example, to change the PAGER variable under C shell, enter:

```
setenv PAGER pg
```

This sets the value of the PAGER environment variable to the command *pg*(1). The PAGER variable is used by *mail*(1).

Bourne and Korn shell users set environment variables like this:

```
$ PAGER=pg ; export PAGER
```

Environment variables can be set on the command line, or in either of the shell startup files */etc/profile* or *$HOME/.profile*.

Configuring Default File Permissions With umask

A system default called *umask* controls the access permissions of any files or directories that you create. The system default for IRIX, without *umask* set, is 022, which sets the following permissions respectively:

user	Full access: read, write, and, if applicable, execute permission. Directories can be executed; that is, you can ''change directories'' into any of your own directories and copy files from them.
group	Anyone in the same group can read and, if applicable, execute other group members' files. Execute permission is turned on for directories. Write permission is turned off.
other	All other people on the system have the same access permissions as group access.

The system default *umask* of 022 is the same as running *chmod 644* on files that you create and *chmod 755* on directories and executable files that you create. Setting your *umask* does not affect existing files and directories. To change the default permission, use the *umask* shell command. Like *chmod*, *umask* uses a three-digit argument to set file permissions. However, the argument to *umask* works the opposite as the argument to *chmod*. The argument to *umask* lowers the access permissions from a maximum of 666 (full access for files) and 777 (full access for directories).

The following command leaves permission unchanged for user, group, and other when you create files and directories:

```
umask 000
```

This command reduces access for other users by 1 (it removes execute permission):

```
umask 001
```

This command reduces access for group by 1 (no execute permission) and for others by 2 (no write permission, but execute is allowed):

```
umask 012
```

This command removes write and execute permission for group and removes all permissions for others:

```
umask 037
```

For more information, see the *umask*(1) reference page.

Configuring Special Login Shells

You may want to assign an account a login shell other than one of the system defaults. Reasons for doing this include:

- The need for special-use accounts that require restricted or very specific access to the system
- a user requesting a special shell

You can specify any program as the login shell for an account. For example, you can use a third-party application program as the login shell. Users with this application as a shell log in to the system and are immediately placed in the application. All interaction with the system is through the application, and when the users quit from the application, they are automatically logged out. To restrict access to the system, you can also use a custom shell that you create.

Another example is the *nuucp* account, which uses */usr/lib/uucp/uucico* as a login shell.

Many users have favorite shells, for example the *bash* shell, that they might want you to install. As with any other software, make sure it comes from a reputable source. (*bash* shell is public domain software .) You may wish to back up the system completely before installing the shell, then monitor the system closely for a while to be sure there are no problems with the shell.

For security reasons, you should not blindly accept compiled binaries and install them as login shells on the system (or anywhere else on the system, for that matter). Start with the source code for the shell, make sure there are no security holes in the code, then compile it for your site.

Note that special shells should be located in a file system that is always mounted before users log in to the system. If the file system that contains a login shell is not mounted, people who use that shell cannot log in to their accounts.

Communicating with Users

There are several ways to communicate with users in the IRIX system, including electronic mail, the message of the day, the remote login message, *news*, *write*, and *wall*.

Electronic Mail

Users can send messages to one another using one of the electronic mail programs provided with IRIX. Media Mail is a graphical mail reader with an extensive on-line help system. Media Mail also has a command line interface for those systems without graphics capability. For a complete discussion of configuring electronic mail, see the *IRIX Admin: Networking and Mail* guide.

Message of the Day

You can communicate items of broad interest to all users with the */etc/motd* file. The contents of */etc/motd* are displayed on the user's terminal as part of the login process. The login process executes */etc/cshrc* (for the C shell), which commonly contains the command:

```
cat /etc/motd
```

Any text contained in */etc/motd* is displayed for each user every time the user logs in. For this information to have any impact on users, you must take pains to use it sparingly and to remove outdated announcements.

A typical use for the message of the day facility might be

```
5/30: The system will be unavailable from 6-11pm Thursday, 5/30 while
we install additional hardware
```

Be sure to remove the message when it is no longer important.

Creating a Message of the Day

The file */etc/motd* contains the "message of the day." This message is displayed on a user's screen, either by */etc/profile* if the user runs Bourne shell, or by */etc/cshrc* if the user runs C shell, when the user first logs in to the system.

You can place announcements of system activity in the *motd* file. For example, you should warn users of scheduled maintenance, changes in billing rates, new hardware and software, and any changes in the system or site configuration that affect them.

Since users see this message every time they log in, you should change it frequently to keep it from becoming stale. If users see the same message repeatedly, they lose interest in reading the message of the day and can miss an important announcement.

Make sure you remove outdated announcements. If nothing new is happening on the system, trim the file to a short "welcome to the system" message.

A typical *motd* file looks something like this:

```
Upcoming Events: ---------------


26 November -- The system will be down from 8PM until Midnight for a
software upgrade. We are installing FareSaver+, Release 3.2.2d.


Watch this space for further details.


28 November through 31 November -- We will be operating with a minimal
staff during the holiday. Please be patient if you need computer
services. Use the admin beeper (555-3465) if there is a serious
problem.
```

The *motd* file is used more frequently on servers than on workstations, but can be handy for networked workstations with guest accounts. You can also send electronic mail to all people who use the system, but this method consumes more disk space, and users may accidentally skip over the mail in their mailboxes.

Remote Login Message

In addition to */etc/motd*, the file */etc/issue* is displayed to whomever logs in to a system over a serial line or the network. If */etc/issue* does not exist, or is empty, no message is displayed.

Creating a Remote Login Message

The file */etc/issue* is displayed before the login prompt is given to someone attempting to log in to a system over a serial line or the network. If */etc/issue* does not exist, or is empty, no message is displayed. This is essentially different from */etc/motd* because it is displayed *before* the login prompt. This message is often used to notify potential users of rules about using the equipment; for example, a disclaimer that the workstation or server is reserved for employees of a particular corporation and that intruders will be prosecuted for unauthorized use.

News

You can set up a simple electronic bulletin board facility with the */usr/news* directory and the *news*(1) command. With *news*, you can post messages of interest about the system. This is not the same system as the publicly distributed Usenet system. Place announcements of interest about the system in the directory */usr/news*. Use one file per announcement, and name each file something descriptive, like "downtime" and "new-network." Use the *news* command to display the items.

You can automatically invoke *news* from a shell startup file, for example from the */etc/cshrc* file. It is a good idea to check for new news items only from a shell startup file, since users may not be ready to read news immediately upon logging in. For example:

```
news -s
```

With the **-s** argument, *news* indicates how many articles there are since you last read news.

When you read news with the *news* command, you can do the following:

read everything

: To read all news posted since the last time you read articles, enter the *news* command with no arguments: **`news`**

select some items

: To read selected articles, enter the *news* command with the names of one or more items as arguments:

```
news downtime new-network
```

read and delete

After you run the *news* command, you can stop any item from printing by pressing **<Ctrl-C>** or **<Break>**. Pressing **<Ctrl-C>** or **<Break>** twice stops the program.

ignore everything

If you are too busy to read announcements at the moment, you can read them later. Items remain in */usr/news* until the administrator (*root*) removes them. The list of news items is still displayed each time you log in.

flush all items

There are two ways to catch up with all current news items:

```
touch .news_time
```

This updates the time-accessed and time-modified fields of the *.news_time* file and thus the *news* program no longer considers there are articles for you to read.

This command prints all current articles, but sends the output to */dev/null* so you do not see the articles:

```
news > /dev/null
```

This brings you up to date without reading any outstanding articles.

Write to a User

Use the *write* command to write messages to a user on the system. For example:

```
write ralph
```

User *ralph* sees this on his screen:

```
Message from root on brooklyn (console) [ Tue Feb 26 16:47:47 ] ...
```

You can wait for *ralph* to respond, or you can begin typing your message. If the other user responds, you see a similar message on your screen.

Type your message. As you press **<Return>**, each line of your message is displayed on the other user's screen.

Usually a *write* session is a dialogue, where each user takes turns writing. It is considered good etiquette to finish your turn with a punctuation mark on a line by itself, for example:

```
I noticed that you are using over 50 meg of disk space.Is there
anything I can do to help you reduce that?
>
```

Entering the greater-than symbol indicates you are through with your paragraph and are waiting for user *ralph* to respond. The other user should choose a different punctuation character to indicate when he is through with his turn.

You can prevent other users from writing to you with *write* by making their terminal or window unwriteable. Use the *mesg* command:

```
mesg n
```

The *n* argument makes your terminal or window unwriteable, and the *y* argument makes it writable. The superuser can write to any terminal or window, even if the user has made his or her terminal unwriteable with *mesg n*.

The *talk*(1) utility is similar to *write*(1), and is preferred by some users.

Write to All Users

The superuser can use the *wall*(1) command to write to all the users who are logged in on the system. This useful when you need to announce that you are bringing the system down.

To use *wall*, enter:

```
wall
```

Enter your message. Press **<Ctrl-D>** when you are finished, and *wall* sends the message.

You can also compose the message in a file, then send it using *wall*:

```
wall < messagefile
```

The *wall* command is not affected by a user's *mesg* setting. That is, a user cannot stop *wall* from displaying on his or her screen. On a graphics display with multiple windows, the message is displayed in all windows.