

CHAPTER 5

Introduction to the Bash Shell

Different Shells

Most UNIX variants allow you to select among several shells. The shell is important to you as a user because it is your window into the system. The many shells available on UNIX variants are similar in that you use them to issue commands, control many aspects of your user environment, write command files and shell programs, and so on. Because you'll probably be spending a lot of time in your shell, you should develop an understanding of several different shells and see whether you develop a preference of one over the other. A particular shell, such as Bash, does not vary much going from one UNIX variant to another. The shells themselves, however, have some unique features. In general, I don't find that users strongly prefer one shell over another. All the shells are similar in functionality and enjoyable to use once you get to know them. Most UNIX variants allow your system administrator to select from among several different shells when configuring users, so he or she usually has some flexibility concerning the shell that users run. In general, system administrators prefer users to

use the same shell making system administration easier in general. Most system administrators, however, will be happy to grant your request for a particular shell if indeed it is available on your system and you have a good reason to use it. I cover the Bash shell in this chapter and the C and KornShell in the following two chapters.

Introduction to Bash

Most UNIX variants allow you to select among several shells. The Linux-based UNIX operating systems I have used configure Bash as the default shell. Bash possesses many of the fine features of other shells, and in fact derives its name from **B**ourne **A**gain **S**hell, which is a dead giveaway that it possesses at least some of the features of the Bourne shell. Bash is similar to other shells in that it provides a user interface to UNIX. You can use the Bash shell in the following three ways:

- Interactively type commands on the command line.
- Group commonly executed sets of commands into command files that you can execute by typing the name of the file.
- Create Bash shell programs using the structured programming techniques of the shell.

These three techniques are listed in the order in which you'll probably use them. First, you log in and use interactive commands. Then you group together commonly used commands and execute them with a single command. Finally, you may want to create sophisticated shell scripts.

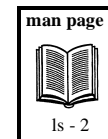
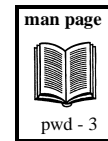
For this reason, describe these aspects of the Bash shell in the order in which they are listed. The command file and programming aspects of the Bash shell are covered as part of the "Shell Programming" chapter. Bash is very similar to the KornShell, which is the

shell used in the shell programming chapter. You can, therefore, use the shell programming chapter as an introduction to programming with Bash as well. Keep in mind, however, that differences always occur when programming with one shell vs. another.

Issuing Commands

The first activity you perform after you log into the system is to issue commands at the prompt. A command you may want to issue immediately is **ls -al**. Here is what I see on my system after executing this command to check my present working directory and producing a long listing of all files when logged in as root:

```
# pwd
# ls -al
total 46
drwxr-xr-x  5 root    root    1024 Nov 26 19:40 .
drwxr-xr-x 20 root    root    1024 Nov  8 20:10 ..
-rw-r--r--  1 root    root     964 Nov 26 19:40 .bash_history
-rw-r--r--  1 root    root     674 Feb  5 1997 .bashrc
-rw-r--r--  1 root    root     602 Feb  5 1997 .cshrc
-rw-r--r--  1 root    root    14815 Nov  8 20:09 .fvmrc.menus.prep
-rw-r--r--  1 root    root     116 Feb  5 1997 .login
-rw-r--r--  1 root    root     234 Feb  5 1997 .profile
drwxr-xr-x  2 root    root    1024 Nov  8 14:10 .seyon
-rw-r--r--  1 root    root     4276 Nov  8 20:09 XF86Config
-r--r--r--  1 root    root    13875 Nov  8 20:05 XF86Config.bak
drwxrwxrwx  2 root    root    1024 Nov 26 19:40 book
drwxr-xr-x  5 root    root    1024 Nov 14 18:12 lg
-rw-r--r--  1 root    root      0 Nov 26 19:40 typescript
#
```

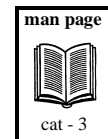


Among the files produced in the long listing of all files is a Bash startup file called **.bashrc**. The following shows the contents of the **.bashrc** file:

```
# cat .bashrc
# ~/.bashrc --
# The individual per-interactive-shell startup file for bash

. /etc/profile

# try solve this tedious 'Backspace vs. Delete' problem...
if [ -z "$TERM" ]; then
  echo ".bashrc: TERM empty: this shouldn't happen!" 1>&2
```



```

echo "   Please contact 'support@lst.de'" 1>&2
else
case $TERM in
linux*)
stty erase '^?'
;;
*)
stty erase '^H'
;;
esac
fi

# general environment settings
#export GROFF_TYPESETTER=latin1
#export LC_CTYPE=iso-8859-1
export LESSCHARSET=latin1
#export METAMAIL_PAGER=less

HISTSIZE=100

alias which='type -path'
alias h=history
alias j="jobs -l"
alias l="ls -Fax"
alias ll="ls -Alg"
alias pd=pushd
alias z=suspend

#

```

Some interesting contents are in the **.bashrc** file. Among them is a value for *HISTSIZE*, which we'll get into shortly, and a set of aliases. These aliases are "shortcuts" for long commands. When I issue the **ll** command, for instance, I am really issuing the **ls -Alg** command.

I may execute both the local **.profile** shown in the earlier long listing as well as **/etc/profile**. **/etc/profile** usually performs setup for all users who log into the system. The following is a listing of **/etc/profile**:



```

# cat /etc/profile
# /etc/profile
# System wide environment and startup programs
# Functions and aliases go in $HOME/.bashrc

PATH="/bin:/usr/bin:/opt/bin:/usr/X11R6/bin:/usr/openwin/bin:/usr/TeX/bin:/usr/
local/bin"

umask 022

if [ `id -gn` = `id -un` ] && [ `id -u` != 0 ]; then
    umask 002
fi

if [ -z "$UID" ]; then
    UID=`id -u`
fi

if [ "$UID" = 0 ]; then
    PATH=/sbin:/usr/sbin:$PATH
else

```

```
    PATH=$PATH:
fi
USER=`id -un`
LOGNAME=$USER

export PATH USER LOGNAME

HOSTNAME=`/bin/hostname`
MAIL="/var/spool/mail/$USER"

export HOSTNAME MAIL

if [ -n "$BASH_VERSION" ]; then
    # (aliases now in $HOME/.bashrc, resp. /etc/skel/.bashrc)
    export PS1="\u@\h \W]\\$ "
    export HISTSIZE=100
fi
#
```

We'll also cover some of the contents of **/etc/profile**.

Initializing the History List in **.bashrc**

The Bash shell can keep a history list of the commands you have issued. If you wish to reissue a command or view a command you earlier issued, you can use the history list.

You can specify any number of commands to be included in the history list. The following line in **.bashrc** sets the history list to 100:

```
set history = 100
```

One hundred commands will be saved in the history list. When you log, out the last 100 commands you have issued are stored in the history list. The next time you log in you can view these 100 commands; however, as you issue commands, the oldest commands fall off the history list. This fact is shown in the following example:

```
# history
 2 more history
 3 ll
 4 cd ..
 5 pwd
 6 cd ..
 7 ll
 8 cd ..
 9 ll
10 ll log
11 cd log
12 more *
13 l
14 ll
15 cd /
16 ll
17 cd
18 XF86Setup
19 XF86Setup
20 startx
21 ll
22 pwd
23 ll
24 ll /
25 XF86Setup
26 ll
27 startx
28 find / -name XF86Config*
29 cp /usr/X11R6/lib/X11/XF86Config.eg .
30 ll
31 XF86Setup
32 XF86Setup
33 startx
34 ll
35 mv XF86Config.eg XF86Config
36 XF86Setup
37 startx
38 shutdown -h now
39 man ls
40 man ll
41 man ls
42 man file
43 lsr
44 man chmod
45 man chmod
46 shutdown -h now
47 pwd
48 ls -l
49 pwd
50 ls -a
51 ls -al
52 pwd
```

```
53 ls -al
54 more .profile
55 more .bashrc
56
57 alias
58 ll
59 pwd
60 script
61 script
62 scrit
63 script
64 more .bashrc
65 more .bashrc
66 ll
67 more .profile
68 ll
69 more .bashrc | grep P
70 more .profile | grep P
71 env
72 more /.profile
73 more /etc/profile
74 more /etc/profile | grep PS
75 find / -name *profile* -print
76 more .bashrc
77 more .bashrc
78 ll /etc/profi*
79 cp /etc/profile /etc/profile.orig
80 vi /etc/profile
81 exit
82 cp /etc/profile.orig /etc/profile
83 history
84
85 exit
86 history
87 history
88 ll
89 ll
90 history
91 ll /etc/profi*
92 ll /etc/profi*
93 more .bashrc
94 history
95 ll
96 cd /root
97 ll
98 history | more
99 history | more
100 exit
101 history
```

Notice in this example that command number 100 is the **exit**, or command to log out, from the last session. Command number 101 is the **history** command I issued immediately upon establishing the next session.

Recalling from the History List

All these commands (**cp**, **more**, **find**, **ll**) are in the history list with their corresponding numbers. You can repeat the last command with **!!**, the 89th command with **!89**, and the last command that started with "m" with **!m**, all of which are shown in the following example:

```
# !!
history
 3 ll
 4 cd ..
 5 pwd
 6 cd ..
 7 ll
 8 cd ..
 9 ll
10 ll log
11 cd log
12 more *
13 l
14 ll
15 cd /
16 ll
17 cd
18 XF86Setup
19 XF86Setup
20 startx
21 ll
22 pwd
23 ll
24 ll /
25 XF86Setup
26 ll
27 startx
28 find / -name XF86Config*
29 cp /usr/X11R6/lib/X11/XF86Config.eg .
30 ll
31 XF86Setup
32 XF86Setup
33 startx
34 ll
35 mv XF86Config.eg XF86Config
36 XF86Setup
37 startx
38 shutdown -h now
39 man ls
40 man ll
41 man ls
42 man file
43 lsr
44 man chmod
45 man chmod
46 shutdown -h now
47 pwd
48 ls -l
```

```

49 pwd
50 ls -a
51 ls -al
52 pwd
53 ls -al
54 more .profile
55 more .bashrc
56
57 alias
58 ll
59 pwd
60 script
61 script
62 scrit
63 script
64 more .bashrc
65 more .bashrc
66 ll
67 more .profile
68 ll
69 more .bashrc | grep P
70 more .profile | grep P
71 env
72 more /.profile
73 more /etc/profile
74 more /etc/profile | grep PS
75 find / -name *profile* -print
76 more .bashrc
77 more .bashrc
78 ll /etc/profi*
79 cp /etc/profile /etc/profile.orig
80 vi /etc/profile
81 exit
82 cp /etc/profile.orig /etc/profile
83 history
84
85 exit
86 history
87 history
88 ll
89 ll
90 history
91 ll /etc/profi*
92 ll /etc/profi*
93 more .bashrc
94 history
95 ll
96 cd /root
97 ll
98 history | more
99 history | more
100 exit
101 history
102 history
# 189
ll
total 44
-rw-r--r--  1 root    root          956 Nov 26 19:33 .bash_history
-rw-r--r--  1 root    root          674 Feb  5 1997 .bashrc
-rw-r--r--  1 root    root          602 Feb  5 1997 .cshrc
-rw-r--r--  1 root    root       14815 Nov  8 20:09 .fvwmrc.menus.prep
-rw-r--r--  1 root    root          116 Feb  5 1997 .login
-rw-r--r--  1 root    root          234 Feb  5 1997 .profile
drwxr-xr-x  2 root    root          1024 Nov  8 14:10 .seyon
-rw-r--r--  1 root    root          4276 Nov  8 20:09 XF86Config
-r--r--r--  1 root    root       13875 Nov  8 20:05 XF86Config.bak
drwxrwxrwx  2 root    root          1024 Nov 13 21:25 book
drwxr-xr-x  5 root    root          1024 Nov 14 18:12 lg
-rw-r--r--  1 root    root           0 Nov 26 19:36 typescript
# !m
more .bashrc
# ~/.bashrc --
# The individual per-interactive-shell startup file for bash

./etc/profile

```

```

# try solve this tedious 'Backspace vs. Delete' problem...
if [ -z "$TERM" ]; then
  echo ".bashrc: TERM empty: this shouldn't happen!" 1>&2
  echo "  Please contact 'support@lst.de'" 1>&2
else
  case $TERM in
    linux*)
      stty erase '^?'
      ;;
    *)
      stty erase '^H'
      ;;
  esac
fi

# general environment settings
#export GROFF_TTYPESETTER=latin1
#export LC_CTYPE=iso-8859-1
[7m--More--(70%)[m
export LESSCHARSET=latin1
#export METAMAIL_PAGER=less

HISTSIZE=100

alias which='type -path'
alias h=history
alias j="jobs -l"
alias l="ls -Fax"
alias ll="ls -Alg"
alias pd=pushd
alias z=suspend

[root@nycald1 /root]#
Script done on Thu Nov 26 19:39:51 1998

```

Table 5-1 includes some of the more commonly used history list recall commands.

Table 5-1 Recalling from the History List

Command	Description	Example
! <i>N</i>	Issue command N	! 2
!!	Issue last command	!!
! <i>-N</i>	Issue N th command from last command issued	! -N
! <i>str</i>	Issue last command starting with str	! c

Command	Description	Example
! <i>str</i> ?	Issue last command that had str anyplace in command line	! cat ?
! <i>str1</i> str2	Append str2 to last command with str1	! cd /tmp
^ <i>str1</i> ^ <i>str2</i> ^	Substitute str2 for str1 in last command	^ cat ^ more ^

Editing on the Command Line

Using the history list is a great way of viewing and reissuing commands. Bash also supports command-line editing. You can use the up arrow key to move back one command in the history list. When you press the up arrow key, the last command from the history list appears on the command line. Every time you press the up arrow key, you move back one more command in the history list. When a command appears on the command line, you can press the "Enter" key to issue the command. You can modify the command by using the left and right arrow keys to move to a point in the command line and type additional information, or use the "backspace" and "delete" keys to remove information from the command line.

Aliases in `.bashrc`

An alias is a name that you select for a frequently used command or series of commands. You can use the `.bashrc` file as a place where your aliases are stored and read every time you log in. In the earlier `.bashrc` file, seven aliases were already set up. You can add additional aliases in the `.bashrc` file or define aliases at the command-line prompt, but these will be cleared when you log out.

Here is a list of the aliases that are already set up for us in the `.bashrc` file and an example of running the aliases **I** and **ll**:

```
# alias
alias h='history'
alias j='jobs -l'
alias l='ls -Fax'
alias ll='ls -Alg'
alias pd='pushd'
alias which='type -path'
alias z='suspend'
#
# l
./                ./                .bash_history    .bashrc
.cshrc            .fvwmrc.menus.prep .login           .profile
.seyon/          XF86Config       XF86Config.bak   book/
lg/              typescript
#
# ll
total 44
-rw-r--r--  1 root   root           970 Nov 26 21:35 .bash_history
-rw-r--r--  1 root   root           674 Feb  5 1997 .bashrc
-rw-r--r--  1 root   root           602 Feb  5 1997 .cshrc
-rw-r--r--  1 root   root          14815 Nov  8 20:09 .fvwmrc.menus.prep
-rw-r--r--  1 root   root           116 Feb  5 1997 .login
-rw-r--r--  1 root   root           234 Feb  5 1997 .profile
drwxr-xr-x  2 root   root           1024 Nov  8 14:10 .seyon
-rw-r--r--  1 root   root           4276 Nov  8 20:09 XF86Config
-r--r--r--  1 root   root          13875 Nov  8 20:05 XF86Config.bak
drwxrwxrwx  2 root   root           1024 Nov 26 19:41 book
drwxr-xr-x  5 root   root           1024 Nov 14 18:12 lg
-rw-r--r--  1 root   root            0 Nov 26 21:35 typescript
#
```

These are all very useful indeed, but let's now set up our own alias. Suppose that we want to know how many processes are running on the system. We'll create an alias called "procs" that does this for us. The `ps` command produces a list of processes. We'll issue `ps` and pipe (`|`) this output to `wc` with the "l" option to tell us how many lines are present. The pipe (`|`) directs the output of `ps` to be used as the input to `wc`. The `ps` command produces a list of processes and `wc -l` gives us a count of the number of lines. Therefore, we'll know the total number of processes running. The following example first shows the output of `ps`, then our `alias` command, and finally the output produced by the `alias` command:



```
# ps
PID TTY STAT  TIME COMMAND
188  2  S    0:00 /sbin/getty tty2 VC linux
189  3  S    0:00 /sbin/getty tty3 VC linux
190  4  S    0:00 /sbin/getty tty4 VC linux
```