12 Unix Primer

Early on in the 1960s and 1970s, every major computer manufacturer supplied operating system as a proprietary software. Such an OS was written specifically for their own machine. In particular, each machine had an instruction set and the operating system was generally written in its intermediate language (often assembly language). As a result, no two operating systems were alike in features. When a user moved to a new machine, he would be expected to learn the new operating system. No two machines could even exchange information, not to mention the notion of portability of software.

It was in this context, that "unics", an acronym for uniplexed information and computing system, was developed by Dennis Richie at Bell Laboratories in USA. The idea was to offer an interpreted common (uniplexed) command language environment across platforms. Unics later became Unix [31].

To implement this idea, Bell Laboratory team developed the idea of isolating a basic "kernel" and a "shell". Most OSs today follow Unix design philosophy of providing a kernel and a shell. Modern Unix-based environments support an extensive suite of tools.

12.1 MOTIVATION

Unix is a popular operating system. Unix philosophy has been to provide a rich set of generic tools and to support tool-based application development. For instance, Unix provides generic string matching tools which are very useful in software development. These tools and utilities also aid in enhancing a user's productivity. Clearly, the main advantage of such an approach is to leave a lot of leeway for the users. Experience indicates that this encourages users to use tools in innovative ways to create new applications. Or they may just create a pleasant customised work environment. Now contrast this with a closed and/or packaged environment. That leaves little room, if any, for creative composition or enhancements. It offers little or no outlet to a user to

customise the working environment. In Unix, users have access to the same tools which are also used by Unix as an OS. This gives one a peek into the working of the internals of Unix as well. In fact, by letting the user enrich the tool suite, or the OS utilities, Unix users expand their horizon. This is what makes Unix an open system.

Besides the tools and utilities orientation, there were two other developments. First, the development of X-windows offered a user a very helpful environment to develop graphical user interface (GUI) for newer applications. Secondly, Unix provided a strong backbone support in the development of computer communication networks by supporting client-server architecture. In fact, the TCP/IP suite of protocols (which forms the core of internet operations) was first developed on Unix.

In this chapter we shall study the elements of the Unix operating systems.

12.2 UNIX ENVIRONMENT

Unix, with X-windows support, provides a virtual terminal in every window. A Unix environment is basically a command driven environment.[1] Each window can be used to give commands. With this a user can run multiple applications, one in each of the windows. A user may open more windows if he needs them (though there is usually an upper limit to the number of windows that may be opened at one time).

Basically Unix provides a user a friendly *shell*. It hides the system *kernel* beneath the shell. The shell interface accepts user commands. The user command is *interpreted* by the shell and then the shell seeks the desired service from the kernel on behalf of the user.

In Unix everything veers around the following two basic concepts:

- Files
- Processes

Users organise applications using files and processes. A program text is a file. A program in execution is a process.

An application may be spread over several files. In addition to program files, there may be other kinds of data files too. Files are generated with some end application in mind. For instance, a file may be a text file for document generation or it may be an image file supporting some medical information in a health-care system. A user may have many applications. Each application may require several files. Unix file system helps users in organising their files. Next we shall study the file system in Unix.

12.3 UNIX FILE SYSTEM

Unix allows a user to group related files within a directory. Suppose we want to work with two different applications, one involving documents and the other involving images.

Maybe we group the document related files under or
another directory. Sometimes it helps to further su
subdirectories to help organise files hierarchically. S
of directories with files as leaves.

When you log into a Unix system, you always d
create files as well as subdirectories starting from

An example of file hierarchy: When I crea
first created a directory COURSES under my hor
created directories called OS, SE, COMPILERS
software engineering and compiler courses. In oth
hierarchy here. The COURSES directory has
COMPILERS within each of which there are module
modules subdirectory there are individual page files.
you to image files or postscript files of the course

It is important to know how one may reach a p
The home directory is usually denoted in Unix by t
particular module in OS course in my organisation c
require that we traverse down from the home to
module. This is called traversing a *path*. Our path
how each step in the path hierarchy is separated b

We say that directory COURSES has a subdirect
also say that COURSES is the parent of directory
a directory called *root*. The root directory has nc
usually a child of a directory named home which is
user's home directories are created with the user'
named bhatt.

The directory where you may be currently locat
The parent of current directory is denoted by tw
symbols. These symbols are used when the user ha
present position in the file system. An absolute p
starting with root as in /home/bhatt/COURSES/OS/mc
COURSES then the same path shall be denoted as ./OS
has no spaces. In fact, no spaces are permitted wit

Unix commands for files: The general co
following structure:

<UnixCommand> <Options> <arguments>

A user may choose one or more options and one o

[2]Unix commands usually have options. Many of the opti

an open system.
tilities orientation, there were two other developments. First,
lows offered a user a very helpful environment to develop
UI) for newer applications. Secondly, Unix provided a strong
elopment of computer communication networks by supporting
n fact, the TCP/IP suite of protocols (which forms the core of
st developed on Unix.
study the elements of the Unix operating systems.

NT

ort, provides a virtual terminal in every window. A Unix
ommand driven environment.[1] Each window can be used to
a user can run multiple applications, one in each of the
nore windows if he needs them (though there is usually an
windows that may be opened at one time).
a user a friendly *shell*. It hides the system *kernel* beneath
accepts user commands. The user command is *interpreted*
ll seeks the desired service from the kernel on behalf of the

around the following two basic concepts:

ns using files and processes. A program text is a file. A
cess.
ead over several files. In addition to program files, there
es too. Files are generated with some end application in
e a text file for document generation or it may be an image
formation in a health-care system. A user may have many
may require several files. Unix file system helps users in
shall study the file system in Unix.

ted files within a directory. Suppose we want to work with
involving documents and the other involving images.

cessible through *man* command. Try *man ls*.

of directories with files as leaves.

When you log into a Unix system, you always do so in your login directory. You may create files as well as subdirectories starting from the login directory.

An example of file hierarchy: When I created course notes for web browsing, I first created a directory COURSES under my home directory. Under this directory I created directories called OS, SE, COMPILERS respectively for operating systems, software engineering and compiler courses. In other words, there is an organisational hierarchy here. The COURSES directory has subdirectories like OS, SE, and COMPILERS within each of which there are modules. At the bottom-most level within the modules subdirectory there are individual page files. The links help you navigate and lead you to image files or postscript files of the course pages.

It is important to know how one may reach a particular file in such a file hierarchy. The home directory is usually denoted in Unix by tilde (~). Suppose we wish to reach a particular module in OS course in my organisation of files as described above. This would require that we traverse down from the home to COURSES to OS to the particular module. This is called traversing a *path*. Our path would be ~/COURSES/OS/moduleX. Note how each step in the path hierarchy is separated by a forward slash.

We say that directory COURSES has a subdirectory (or a child directory) OS. We may also say that COURSES is the parent of directory OS. In Unix, all paths emanate from a directory called *root*. The root directory has no parent. A user's login directory is usually a child of a directory named home which is a child of / (i.e. root). Under home, a user's home directories are created with the user's login name. My home directory is named bhatt.

The directory where you may be currently located is denoted by a period (.) symbol. The parent of current directory is denoted by two periods (..). These two are special symbols. These symbols are used when the user has to make a reference *relative* to the present position in the file system. An absolute path name would trace the file path starting with root as in /home/bhatt/COURSES/OS/module5. Suppose we are in directory COURSES then the same path shall be denoted as ./OS/module5. Note that the file path name has no spaces. In fact, no spaces are permitted within the file path names.

Unix commands for files: The general command structure for Unix[2] has the following structure:

<UnixCommand> <Options> <arguments>

A user may choose one or more options and one or more arguments.

---

[2]Unix commands usually have options. Many of the options such as –i with *rm* command are very useful.

It is important to be well versed with the commands that support the identification of files and navigation on the directory tree. Here is a brief list of relevant Unix commands.

ls	:	Lists all the files within a directory.
cd	:	By itself it brings you back to home directory.
cd pathname	:	Takes you to the directory described by the *pathname*.
rm filename	:	Removes file *filename* from the current directory.
pwd	:	Prints the name of the current working directory.
mkdir subdirname	:	Creates a subdirectory under the current directory with the name *subdirname*.
rmdir subdirname	:	Removes a subdirectory under the current directory with the name *subdirname*.
touch filename	:	Creates a file in the current directory with the name *filename*. This file, on creation has 0 characters and 0 lines.

For now you should use the above listed commands to create files and directories. You should basically learn to navigate the directory tree. Later, we shall learn more commands applicable to files. You should also consult the on-line manual for options available on *ls* and *rm* commands.

- *Ordinary files:* These are usually text files. Programs written in specific programming languages may have specific extensions. For instance, programs written in C have a .c extension. Files prepared for TeX documentation have a .tex extension. These are usually the files users create most often using an editor.
- *Directories:* Subdirectories are treated as files by Unix.
- *Binary files:* These are executables.
- *Special files:* Special files provide input/output capability for Unix environment. Unix treats the device IO as file communication. In fact, this gives the look and feel of a file read or write whenever a user communicates with an IO device.

In general, files have three basic operations associated with them. These are *read*, *write*, and *execute*. Unix supports four types of files.

12.4 SOME USEFUL UNIX COMMANDS

We give here a list of some commonly used Unix commands:

bc	:	Gives you a basic calculator. Try simple arithmetic to get a feel for its operation.
cal	:	Shows the calendar for the current month. A variation of
clear	:	Clears the screen.
cp filename1 filename2	:	Creates a copy of file *filename*
date	:	Shows the current time and da
echo sometext	:	Echos back *sometext* on the te
history	:	Shows the previously given com customised.
more filename	:	Shows the file *filename* one pa same except that it gives scrol
cat filename	:	Displays the file *filename* on t
cat filename(s) > newfile	:	Combines all the files in *filenar* a file *newfile*.
man AUnixCmd	:	Shows the description of the c the on-line help manual. It desc with all its possible options an
exit	:	Exits the current shell.

12.5 UNIX PORTABILITY

In Section 12.1 we described why Unix may be regarded as a main outcomes of such an open system, as Unix has been, is th of Unix with mutual incompatibility. This led to problem interoperability. Fortunately, there are some standards now. standards are X/open and POSIX. In developing the POSIX, like IEEE (URL: http://standards.ieee.org) have been involved. M ensure that their products are POSIX compliant. POSIX com some interface protocols. This ensures interoperability. A very both X/Open and POSIX is http://www.starlink.rl.ac.uk/star/docs/sun121.h

Finally: We discussed some Unix commands in this cha from complete. For instance, Unix allows the use of regular exp have not discussed that in this chapter. This is because in th discussing tools for pattern matching which requires extensive However, some exercises in this chapter may contain regular that the users of Unix are motivated to learn a few such det

EXERCISES

The exercises in this chapter are sometimes beyond the mate chapter. The main reason for this is that this book is not a reader should get into the habit of reading up from other avail

...ists all the files within a directory.

...By itself it brings you back to home directory.

...Takes you to the directory described by the *pathname*.

...Removes file *filename* from the current directory.

...Prints the name of the current working directory.

...Creates a subdirectory under the current directory with the ...ame *subdirname*.

...emoves a subdirectory under the current directory with the ...ame *subdirname*.

...reates a file in the current directory with the name *filename*. ...his file, on creation has 0 characters and 0 lines.

...he above listed commands to create files and directories. You ...gate the directory tree. Later, we shall learn more commands ...d also consult the on-line manual for options available on it.

...se are usually text files. Programs written in specific ...es may have specific extensions. For instance, programs ...extension. Files prepared for TeX documentation have a .tex ...sually the files users create most often using an editor.

...ories are treated as files by Unix.

...e executables.

...iles provide input/output capability for Unix environment. ...IO as file communication. In fact, this gives the look and ...ite whenever a user communicates with an IO device.

...e basic operations associated with them. These are *read*, ...rts four types of files.

COMMANDS

...mmonly used Unix commands:

...Gives you a basic calculator. Try simple arithmetic to get ...a feel for its operation.

...Shows the calendar for the current month. A variation of ...*al* will allow you to view calendar for any month or year.

date : Shows the current time and date.

echo sometext : Echos back *sometext* on the terminal.

history : Shows the previously given command history. This may be customised.

more filename : Shows the file *filename* one page at a time. *less* does the same except that it gives scrolling additionally.

cat filename : Displays the file *filename* on the screen.

cat filename(s) > newfile : Combines all the files in *filename(s)* and outputs to create a file *newfile*.

man AUnixCmd : Shows the description of the command *AUnixCmd* from the on-line help manual. It describes the use of a command with all its possible options and arguments.

exit : Exits the current shell.

12.5 UNIX PORTABILITY

In Section 12.1 we described why Unix may be regarded as an open system. One of the main outcomes of such an open system, as Unix has been, is that there are many versions of Unix with mutual incompatibility. This led to problems in communication and interoperability. Fortunately, there are some standards now. Two of the better known standards are X/open and POSIX. In developing the POSIX, professional organisations like IEEE (URL: http://standards.ieee.org) have been involved. Nowadays most developers ensure that their products are POSIX compliant. POSIX compliant software adheres to some interface protocols. This ensures interoperability. A very good URL which describes both X/Open and POSIX is http://www.starlink.rl.ac.uk/star/docs/sun121.htx/node9.html.

Finally: We discussed some Unix commands in this chapter. This description is far from complete. For instance, Unix allows the use of regular expressions in arguments. We have not discussed that in this chapter. This is because in the next chapter we shall be discussing tools for pattern matching which requires extensive use of regular expressions. However, some exercises in this chapter may contain regular expressions. It is expected that the users of Unix are motivated to learn a few such details on the fly.

EXERCISES

The exercises in this chapter are sometimes beyond the material actually covered in the chapter. The main reason for this is that this book is not a Unix Manual. However, a reader should get into the habit of reading up from other available sources like Unix man

pages. The Unix documentation, man pages, is very thorough and is available on-line. In the beginning one feels somewhat inconvenienced. However, one soon realises the merits of such on-line manual reading. Unix follows very structured documentation practice. So reading Unix man pages is part of learning Unix!! On to the exercises now.

1. Where did Unix originate? Who were its initial authors? What was their basic design philosophy? Which are the two major locations where most of the Unix developments happened? Why is Unix considered an *open system*?
2. Which are the default IO devices in Unix? How are these identified? How does it help to think of communication with IO as file read/write operation? What is the classification of files in Unix? How is input/output redirection done in Unix?
3. Is Unix a time sharing system?
4. Learn to use vi, emacs text editors.
5. What is a basic tool? Give three examples of tools you use often.
6. What does the tilde symbol ~ represent? Similarly, which directories do the period symbol . and two periods .. represent?
7. What is a Unix pipe? Give two examples using pipes.
8. What is a *path* in Unix? What are relative and absolute paths?
9. Choose a directory in which there is no file having substring New in it anywhere. Now in the dotted lines in the questions below, respond to the commands issued in the following session:

```
> pwd
> .....
> cat NewFile
This is new file. It is a test file for this example. ^d
> ls NewF*e
> .....
> mv New*F*e ../
> .....
> ls
> .....
> cd ../; rm NewFile
> ls .....
```

10. What is the difference between Unix commands more and less?
11. How will you view the top (or bottom) 10 lines of a text file?
12. How will you use the wc command to count the number of lines in a text file?
13. How will you list files in the reverse order of their time of creation?
14. Given a directory, how will you find the number of subdirectories it has?
15. How will you find the location of the commands rmdir?
16. Create a text file nfile and copy it in another file called ofile. N in nfile and store it. What command can be used to identify t files nfile and ofile?
17. How will you change the prompt on your terminal?
18. What is a process in Unix context? What is a *login* process
19. What are the options a, d, r, s and x used for, with the ps co
20. What is the use of tr command in Unix?