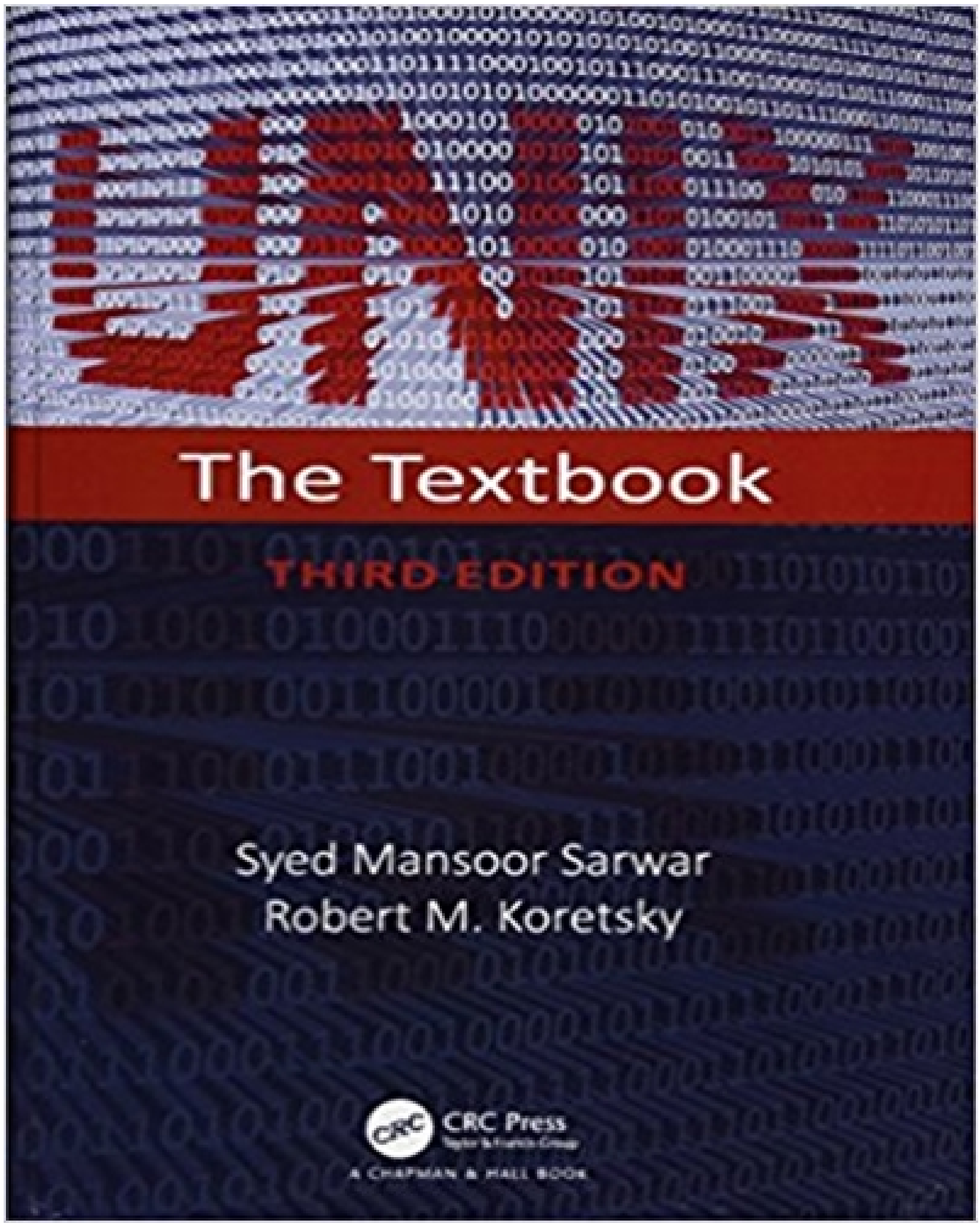
Solutions for UNIX The Textbook 3rd Edition by Sarwar



Solutions

Chapter 2
A “Quick Start” into the UNIX Operating System

1. `mkdir ~/UNIX`

2. You can use the

```
cat lab1 lab2 lab3 lab4
```

command for displaying the four files lab1, lab2, lab3, and lab4. The following commands can be used to perform the same task.

```
cat lab[1-4]
```

```
cat lab[1,2,3,4]
```

You can use the following command to display the lab1.c, lab2.c, lab3.c, and lab4.c files.

```
cat lab[1-4].c
```

You can use the `more` command instead of the `cat` command to display files one page at a time.

3. I used the following command to perform the task.

```
lpr -Pupmpr memo*.ps
```

You can also use the following command to perform the same task.

```
cat memo*.ps | lpr -Pupmpr
```

4. Bourne shell version: `alias w='who -H'`

(This command did not work under Bourne shell on our SunOS based system, but did work under Korn shell on the same system. The command also worked under RedHat LINUX running Bash.)

C shell version: `alias w 'who -H'`

I would put the Bourne shell version in the ~/.profile or /etc/profile file and the C shell version in the ~/.login file for the alias to be effective every time I logon.

I would put the Bourne shell version in the ~/.profile file and the C shell version in the .cshrc file for the alias to be effective every time I start a new shell process.

5. The following command can be used to perform the above task. If you have a System V based UNIX system, you will use the lp command in stead of the lpr command.

```
lpr –Pwpr -#2 ~/UNIX/ls.man for PC-BSD 10.0
```

```
lp –Pwpr -n2 ~/UNIX/ls.man for Solaris 11.1
```

6. Type `mesg` on the command line to find if it is on or off, type `mesg n` to turn it off, and put `mesg n` in your .bashrc or .cshrc file to permanently have it turned on or off for Solaris 11.1 or PC-BSD 10.0.

7. This command uses metacharacters to print on the qpr printer all the files in the current directory that start with a digit and have a .jpg extension.

8. No answer required.

9. `ls -la *.exe`

`ls` -> command, `-la` -> options, `*.exe` -> command argument

```
lpr –Pwpr
```

`lpr` -> command, `-P` -> option, `wpr` -> option argument

```
chmod g+rwx *.*
```

`chmod` -> command, `g+` ->option, `rwx` -> option arguments, `*.*` -> command argument

10. The Description part of each man page is most useful and descriptive for both beginner and intermediate user. The options which are most useful are very subjective and user-dependent, and many (but not all) of them are only useful for advanced users that are doing shell programming or concatenating multiple UNIX commands with output redirection.

11. Ten users are currently logged on our system. I used the `who` command and counted the number of lines for the unique users (some users were logged on more than once) who are currently logged on. There are other ways to determine the number of users currently logged on your system; we discuss them in Chapter 9 by using other shell commands and UNIX pipes.

12. I used the `uname` command to determine the answer to the question. The `uname -a` command displays full information about your system, including kernel version, CPU, etc.

13. The following command performs the task on both PC-BSD and Solaris 11.1.

```
man s2 socket read connect
```

14. When you log on, the UNIX system starts running a program that acts as an inter-face between you and the UNIX kernel. This program, called the UNIX *shell*, executes the commands that you have typed in via the keyboard. When a shell starts running, it gives you a prompt and waits for your commands. When you type a command and press <Enter>, the shell interprets your command and executes it. If you type a nonexistent command, the shell so informs you, redisplays the prompt, and waits for you to type the next command. Because the primary purpose of the shell is to interpret your commands, it is also known as the UNIX **command interpreter**.

15. A shell command can be internal/built-in or external. The code to execute an internal command is part of the shell process, but the code to process an external command resides in a file in the form of a binary executable program file or a shell script. We describe in detail how a shell executes commands in Chapter 10.

16. The names of five UNIX shells are: Bourne (sh), Bourne Again (Bash), Korn (ksh), C (csh), TC (tcsh), rc (rc), and Z (zsh). The most popular shells are Bash, , C, Bourne, Korn, and TC shells. Our login shell is the C shell on PC-BSD 10.0, and Bash shell on Solaris 11.1.

Because the shell executes commands entered from the keyboard, it terminates when it finds out that it cannot read anything else from the keyboard. You can so inform your shell by pressing <Ctrl+D> at the beginning of a new line. As soon as the shell receives <Ctrl+D>, it terminates and logs you off the system. The system then displays the login: prompt again, informing you that you need to log on again in order to use it.

To terminate or leave a non-login shell and return to your default login shell, you should press <Ctrl+D> on a blank line. If this way of terminating the new shell doesn't work, type exit on the command line and then press <Enter>. By doing so, you halt the running of the new shell, and the default shell prompt appears on your display.

17. The Bourne Again and Korn shells are supersets of the Bourne shell, and the TC shell is a superset of the C shell.

18. The directories that a shell searches to find the file corresponding to an external command comprise the search path for the shell. The directory names are stored in the shell variable *PATH* in the Bourne, Korn, and Bash shells and *path* in the C shell. Directory names are separated by colons in the Bourne, Korn, and Bash shells and by spaces in the C shell. You can view the search path for your variable by using the echo *$PATH* command in the Bourne, Korn, and Bash shells and the echo *$path* command in the C shell.

The search path variables are typically stored in the initial system startup files, usually .profile for Solaris 11.1 and .login for PC-BSD 10.0, in your home directory. This causes the search path for your login shell to be initialized when you login. You can put these variables in the shell start-up files also, which are .csh for the C shell and .bash for the Bash shell.

19. You can use the echo $PATH or echo $path command to determine the search path for your environment. The search path in our environment is shown in the output of these commands, as shown below.

```
$ echo $PATH
/usr/sbin:/usr/X11/include/X11:.:/users/faculty/sarwar/bin:/usr/ucb:/bin:/
usr/bin:/usr/include:/usr/X11/lib:/usr/lib:/etc:/usr/etc:/usr/local/bin:/u
sr/local/lib:/usr/local/games:/usr/X11/bin
$
```

```
% echo $path
/usr/sbin /usr/X11/include/X11 . /users/faculty/sarwar/bin /usr/ucb /bin
/usr/bin /usr/include /usr/X11/lib /usr/lib /etc /usr/etc /usr/local/bin
/usr/local/lib /usr/local/games /usr/X11/bin
%
```

You should run the following command so that your shell searches your current as well as your ~/bin directories while looking for an external command that you type.

```
$ PATH=$PATH:~/bin:.
$
```

The order of search is: currently set search path, followed by your ~/bin directory, and then your current directory.

20. A hidden file is one which is not displayed as part of the output of the ls command. These files are also known as the dot files because their names start with a dot as in .bashrc. The names of the hidden files on Solaris 11.1 and PC-BSD 10.0 UNIX are .profile and .login, respectively. These files are also known as system startup files.

21. A shell startup file executes when you start a shell process. The name of this file is .cshrc for the C shell. All startup files are stored in your home directory.

22. Do it yourself.

23. Let's assume that the Bourne shell script is in a file called my_script. In order to execute the script in this file, I will take the following four steps.

1. Make the script executable by executing the chmod u+x my_script command. (See Chapter 8 for file permissions.)
2. Start a Bourne shell process on top of my login C shell.
3. Execute the shell script by running the my_script as a command.
4. Terminate the Bourne shell process and get back to my login C shell.

The following shell session show the four steps.

```
% chmod u+x my_script
% /usr/bin/sh
$ ./my_script
[ Output of the script ]
$ <Ctrl+D>
%
```

Another way to execute the shell script is to make it executable, put #!/bin/sh on the first line of my_script, and execute the script.