

CHAPTER 6

Introduction to Unix

The Unix operating system is an incredibly powerful and complex system that is ideal for running a distributed system such as ours, particularly since we use computers primarily for program development. A wide variety of tools are available for doing the kinds of work we do and the ability to have a large number of users communicate with each other is built in.

File Systems

Our system has one very large and complicated hierarchical file system, that is, the files are organized into an inverted tree structure with a root directory at the top of the structure and then a set of subdirectories from the root directory and these subdirectories themselves have subdirectories, and so on. At the root of the system are “system files,” that is, files needed to operate the system. Down a couple of levels from the root directory is a directory containing subdirectories `grad`, `stat`, and `staff`. These directories have subdirectories for each graduate student, each faculty member, and each staff member, respectively. Thus each user in the system has their own subdirectory (called their “home directory.”) It is possible to organize a set of users into a “group.” Most users try to organize their files into a set of subdirectories within their home diectory so they can be easily found.

Permissions

Each file in the system has an owner (typically the user who created the file) and a set of permissions attached to it, that is information on who can read, modify, or execute that file (if it is executable). Entering the command `ls -l` will show a list of files and the attached permissions. This list will look something like the following

```
drwx----- 2 jnewton      512 Sep  5 11:33 exams
-rw-rw-r--  1 jnewton      305 Aug 26 09:01 hald.dat
-rw-rw-r--  1 jnewton    32959 Sep  5 11:19 introsys.tex
-rwxrwxr-x  1 jnewton   204800 Aug 24 10:18 regswp
-rw-rw-r--  1 jnewton     8271 Aug 24 10:18 regswp.f
```

Each line has the permissions followed by the owner, the size of the file, the date and time it was last modified, and finally the name of the file. There are 10 fields in the permissions. The first is a `d` if the entry is a directory. The remaining nine entries are in three groups of three each with the three groups being owner, group, and world. Thus `jnewton` can read, write, and execute the directory `exams` and no one else can, everyone can read `hald.dat`, `introsys.tex`, and `regswp.f`, while `regswp` is an executable file that anyone can execute. Note that only `jnewton` and his group can modify all these files (they have write permission as seen by the `w` in the permissions).

To change the permissions on a file (or a directory) belonging to you, issue the command `chmod xyz`

`filename` where `x`, `y`, and `z` are each numbers between 0 and 7 summarizing the permissions for owner, group, and world. They are the sum of the read (4 points for yes, 0 for no), write (2 points for yes, 0 for no), and execute (1 point for yes, 0 for no). Thus to totally protect a directory, you can issue the command `chmod 700`.

Man Pages

Briefly documenting (or remembering) all the commands available in Unix (and all their options) is impossible. Thus Unix supplies the command `man` which will display the manual pages for each command on the system. For example, `man ls` will show you all you ever wanted to know about the `ls` command (and probably more than you want to know.)

Commonly Used Commands

Command	Purpose
<code>cd</code>	Change directory
<code>mkdir</code>	Create a new directory
<code>ls</code>	List the files and directories in the current directory (short version)
<code>ls -l</code>	Long version
<code>cat</code>	List the contents of a file or concatenate files
<code>more</code>	List the contents of a file one screenful at a time
<code>grep</code>	Display all lines in a file containing a certain character string
<code>awk</code>	Perform operations on specified lines in a file
<code>wc</code>	Show the number of lines, words, and bytes in a file
<code>cut</code>	Select from a file certain fields or sets of columns
<code>paste</code>	Horizontally concatenate files
<code>lpr</code>	Print a file
<code>lpq</code>	See what's printing on a specified printer
<code>sort</code>	Sort a file
<code>uniq</code>	Find unique lines in a file
<code>ps</code>	See what jobs you are currently executing
<code>kill</code>	Kill a job that is executing

Redirection and Pipes

Redirection is the idea of using a file as input to a program that expects something from the keyboard or sending output to a file instead of the screen. For example `sort fname > sfname` will sort the file called `fname` and redirect the output to the file named `sfname` instead of to the screen. To concatenate the output to the end of a file, used two greater than signs, for example, `sort fname >> sfname` will append the result of sorting `fname` to the end of `sfname`.

One can connect a series of commands on a single command line using pipes, for example

```
grep alias .mailrc | sort | uniq > new.mailrc
```

will pull out all lines in the `.mailrc` file having the string `alias`, then sort them, then eliminate any duplicates, and then send the result to a file called `new.mailrc`.

Some Examples

I have a set of files called `fa197.ros`, `spr88.ros`, `sum88.ros`, and so on, up to `sum94.ros`, where the lines in a file consist of information about each student who took a statistics course during that semester, including: 1) their ID in columns 1–9, 2) their name in columns 10–41, 3) their classification in 42–43, 4) their major in 44–47, 5) the course number in 48–50, and 6) the section number in 51–53. Thus the following

line (which I've displayed on two lines so it will fit)

```
awk '{print substr($0,48,3) substr($0,44,4)}' spr94.ros | sort |
  uniq -c > bymaj
```

will form a file containing a list of all majors (and how many in each major) taking each course from the statistics department in the Spring of 1994.

Aliases and Shell Scripts

If you create a file called `.alias` in your home directory with lines like

```
alias 604 'cd /tex/classes/60494'
```

and then say `source /.alias`, you can use the aliases in the `.alias` file.

You can also create a file called a shell script with one or more unix commands that can be executed as though it were a unix command itself. For example, if I create the file called `texprt` containing

```
tex $1
dvips -Php1 $1
```

and then say `chmod +x texprt`, then I can `TEX` and print (on `hp1`, see below) a file by saying `texprt filename`, where `filename` is the name of the `TEX` file.

Printers

The network has two public printers; one in room 421 (named `hp1`) and one in the main office (named `hp2`). To print a file on one of these printers, enter the command `lpr -Pprintername filename` where `printername` is either `hp1` or `hp2`. Graduate students should never print anything longer than a few pages unless they have permission from Drs. Newton or Schmiediche. In particular, do not print documentation or books!

When printing computer code (or any ASCII file), please use the Unix program `a2ps`. This program prints the code sideways on the paper in two columns. A total of 132 lines can be printed on one page and still be very readable. Use the command `a2ps -Pprintername filename` where `printername` is the name of the printer you want the file sent to.

Using the System via a Modem

See Dr. Schmiediche.

The emacs Text Editor

Almost everything one does on the system requires the use of a text editor, including writing C or Fortran programs, using `TEX` and `Splus`, and just doing ordinary operating system activities. Almost all Unix systems have a text editor called `vi` (see `man vi`) but the editor of choice among most users is `emacs`, a very sophisticated editor with many features.

Starting emacs

Typically, when one logs into the system from a workstation or X-terminal, one should open an emacs window by entering the command `emacs &` (the ampersand tells Unix to run emacs in the background, that is, allow you to do other things while the emacs window is there.)

The Control and Meta Keys

While in emacs, if you type a series of characters they will appear on the screen. Thus one needs a way to tell emacs that what is to be typed next is not text but a command to emacs. There are two keys that signal to emacs that what comes next is a command. These are called the “control key” (the key with the word control on it) and the “meta key” (which key this is on a keyboard varies from keyboard to keyboard; on our X-terminals it is the key marked “Alt” on the keyboard.)

These keys are used typically by holding it down and then typing some other key. For example, while in emacs, one tells emacs to begin editing a file by typing `Ctrl-x-f`, that is, holding down the control key and striking the x and f keys. Then emacs will ask you for the name of the file to edit (the question will appear in the blank line at the bottom of the screen called the “minibuffer”) and you answer with the name of the file.

Buffers

One can edit several files simultaneously. Each file is contained in a “buffer,” and one buffer is active at all times.

Editing Commands

Like any other full-screen text editor, one needs to know only a few things in order to edit a file. These things include:

Keystroke	Meaning
Moving the cursor	
arrow keys	Up or down one line or left or right one character
meta-v or Ctrl-v	Up or down one screen
meta-< or meta->	Beginning or end of file
meta-x-goto-line	Go to a specified line number (emacs prompts for number)
Deleting things (see block operations for deleting a block)	
Ctrl-d	Delete the current character
Ctrl-k	Delete to the end of the current line
Searching and replacing	
Ctrl-s	Search for character string (emacs prompts for string)
meta-%	Global search and replace emacs will prompt for old and new strings and prompt yes or no for each occurrence; answer with ! to replace all remaining occurrences
Block operations	
Ctrl-@, Ctrl-w	Mark a set of lines (Ctrl-@ at start and Ctrl-w at end) and delete that set
Ctrl-@, meta-w	Mark a set of lines but don't delete that set
Ctrl-y	Copy a marked set of lines to current cursor position
Miscellaneous	
Ctrl-x-s	Save the file in the current buffer

`Ctrl-x i`

Insert a file where the cursor is (emacs will prompt for file name)

Introduction to email

The Unix `mail` program can be used to either read mail or send mail.

Sending Mail

To send a message to someone, you type `mail address` where `address` is the address (or list of addresses) of the person (or persons) to receive the mail. The `mail` program will then ask you for a subject, which you enter. Then `mail` will ask you if you want carbon copies sent to anyone (by giving you a `CC:` prompt). If you want to, you can enter an address (or a list of addresses) of the person (or persons) to receive the carbon copy. If you don't want carbon copies, just hit enter in response to the `CC:` prompt. Then you type your message, hitting the enter key at the end of each line. When you have finished the message, you type `Ctrl-d` when the cursor is at the beginning of a blank line.

To email a file, enter the command `mail address < filename`.

Reading Your Mail

When mail for you comes to our system, it goes into a file called

```
/var/spool/mail/acct
```

where `acct` is the name of your account. This file is called your "mail spooler." If you enter the command `mail` from the Unix prompt, Unix looks to see if there is anything in your mail spooler. If not, it gives you the message, `no mail for acct` where again `acct` is the name of your account. If your spooler is not empty, `mail` reads it, puts a summary of its contents on your screen, and then issues a prompt (`mail&`). The summary will look something like

```

U 1 rondoes@fwi.uva.nl Fri Jul 22 01:28 42/1403 Test
U 2 ruth                Wed Aug 3 10:25 15/521 PPRI
U 3 gerig@stat.ncsu.edu Wed Aug 3 12:35 20/824 Sign On
U 4 newsl                Wed Aug 3 12:51 19/498 Newsletter

```

that is, the line for each message contains the number of the message as well as the address of the sender and the date and time it was received and finally the subject of the message.

If you enter the number of a message, its contents will be displayed on the screen. You can delete a message from the spooler (by `d number`), or save a message to a file (by `s number filename`), or get a list of all the things you can do (by `?`).

Leaving the Mail Program

When you want to return to the Unix prompt, you have two choices of how to leave `mail`. The first is to type `x` in which case your spooler will be preserved exactly how it was when you entered the `mail` program. If you enter `q`, then any messages you read or deleted will be removed from the spooler. If you have read any messages and exit `mail` with a `q`, then the contents of the read messages will be appended to a file called `mbox` that is in your home directory (this is called your "mbox.") To look at the contents of your `mbox`, enter the command `mail -f` from the Unix prompt.

Mail Aliases

In your home directory is a file called `.mailrc`. In this file you can create mail aliases such as

```
alias scott          scott@stat.rice.edu
alias wegman        ewegman@endor.galaxy.gmu.edu
alias kettenring    jon@bellcore.com
alias board         scott wegman kettenring
```

There are also a series of “system mail aliases,” that are available for all users to use, including `all`, `grads`, `faculty`, `staff`, and so on.

Folders

If you put the line `set folder=.Mail` in your `.mailrc` file and create the directory `.Mail` in your home directory, then you can use what are called “mail folders.” For example, you might like to save all email messages for a particular class in a folder having the name of the class. You do this by `s number +folder` from the mail prompt, where `number` is the number of the message and `folder` is the name of the folder. To look at the messages in a particular folder, enter the command `mail -f +folder` from the Unix prompt, where again, `folder` is the name of the folder.