Brock University Physics Department
St. Catharines, Ontario, Canada L2S 3A1

Introduction to Computing at Brock for Physics Students

Edward Sternin

Acknowledgements

In putting together a document such as this, one consults a number of sources. The original outline and much of the material for several of the core sections is from *An Introduction to Unix at USC* by Jon Nightingale, published by the University Computing Services at the University of British Columbia. That document, in turn, had borrowed extensively from the documents created at the University of Southern California. Several key Brock–specific items were contributed by Mike Grau, the System Manager of our Unix servers. Several later sections were inspired by the documentation generated at the Tri–University Meson Facility (TRIUMF).

A considerable typing and editing effort by Alice Witvoet is gratefully acknowledged.

Contents

Chapter 1

Introduction

In 1989, Brock University's Senate declared, "as one of the University's goals, that by year 2000 all students graduating from Brock University will know how and when to use computers in their selected disciplines".

1.1 About this document

The above declaration means that your education will not be complete until you learn how to apply the modern computer tools to your chosen profession. This is especially important in the Sciences, where computers have become so indispensable.

Working towards this goal, the Physics Department has prepared the document you are holding in front of you. It is admittedly limited in scope, serving only as a basic introduction to the computing environment at Brock. However, it's a good starting point if you have had limited previous experience with computers. Once you "get your feet wet", go on to other sources of information: use books, ask experts, and above all, try things out! The computers will forgive you your errors, and may even guide you along through computer–based tutorials, help files, and on–line documentation. Some additional references and sources of information are listed in Chapter 11.

Scientific computing often means Unix, which is the name of a popular operating system (the main piece of software controlling everything else a computer does), and many different computers run variants of Unix. This includes our Departmental server, called `bohr`. You may go to B228 to visit `bohr` in person, but most of the time you will be interacting with it through different terminals located in many computer labs and offices on campus. Almost all of them are "on the network", *i.e.* connected together through a high–speed communications link. You can establish a connection to `bohr` in any number of ways, and you may need some additional help on how to do that from a particular terminal or personal computer, beyond the basic information contained in this manual. We also assume here that you already know the name of your account, password and the name of the computer on which the account exists (for most of you, it is `bohr`). The steps required to obtain all this information are *not* covered here.

This manual *does* cover the following topics:

- logging in and out of a Unix system;
- navigating among files and directories;
- changing your computing environment;
- generating printed output;
- getting help;
- using some software packages.

1.2 Some basics

The Return key

Throughout this document reference is made to commands that should be typed at the keyboard. Make sure that all commands are followed by a Return. That is, when you've typed a command, press the key labelled "Return". The key may also be labelled "Enter" on some keyboards. Pressing this key tells the computer that it should carry out what was typed at the keyboard. Typing a command followed by a Return is called entering a command.

Syntax

We tried to provide a generous number of examples throughout this document. With examples shown as

`cp [-ip]` *filename1 filename2*

the characters inside of the `[ ]` are meant to be optional arguments. The `[ ]` themselves are not typed — only the arguments inside of the brackets, and only if you want to use that option.

The dots `...` in a command indicates that more than one may be specified. For example,

`rm` *filename ...*

means that more than one filename may be listed here, separated by blanks.

Case sensitivity

The Unix operating system is case–sensitive when it interprets commands. That is to say, upper and lower case letters each mean something different to the operating system. Most commands, options, and file names use only lower case, with upper case occurring occasionally. Be careful to use upper and lower case exactly as shown in examples.

Control key

The Control key is similar to the shift key on a typewriter. It is labelled "Control", "Ctrl", or something similar on the keyboard. It is used to shift the keyboard into control mode. Like the shift key, the Control key must be held down while you press another key. Upon releasing the Control key, the keyboard is returned to its normal state. In this document the symbol "Ctrl-" is used to represent the Control key. Thus Ctrl-c means: hold down the Control key and press the c key.

The shell

The Unix process that receives and interprets the commands you type is called a shell. The shell is described in more detail in Chapter 8. The shell lets you know that it is ready for you to enter a command by displaying a shell prompt on your screen. The default prompt on most systems at Brock takes the form

`bohr nn %`

where `nn` is the number of commands issued since command tracking began, and `bohr` is the hostname. For the sake of simplicity, the single character

`%`

represents a Unix prompt throughout this document. When entering a command to the shell, you should *not* type the prompt itself.

Chapter 2

Logging in, out, and setting passwords

This Chapter is about logging into (i.e. gaining access to) and logging out of (disconnecting from) Unix, and setting your password. Maintaining the integrity of your account depends largely upon your password. The first thing you should do if this is your first time logging in is to change your password. This is important because the current password is probably written down somewhere, or someone might have overheard it when the account was set up.

2.1 Getting started with telnet

There are numerous ways to access the Unix systems here on campus. We will describe here how to access `bohr`, the procedure is very similar for other Unix hosts. Telnet is the name of a common network communications protocol; telnet–style access is possible from any personal computer (PC or Mac) connected to the campus network. This is the basic form of connection, usually appropriate for using the host computer in text mode only. Your personal computer becomes a terminal connected to the remote host. The details will depend on the type of the computer you are using, so if you get stuck you may need to consult the manuals and/or ask someone who knows about that particular computer. Below is just a very basic outline of several most popular ways of accessing `bohr`:

NCSA Telnet NCSA (National Center for Supercomputing Applications) software is free, and that is why it may be the single most popular network software on campus. It exists for both IBM PCs and Macs, and allows you to open up one or several telnet connections to the same or even to several different Unix hosts.

Typically, you would enter a command like this one:

```
C:\> telnet bohr
```

(in DOS mode) or double–click on an icon named "Telnet" (in Windows mode, or on a Mac) and then enter `bohr` when prompted for a host name. There may also be an icon named "bohr"; usually all you need to do is to double–click on it.

Brock University also has a site license for a commercial package called PC/TCP made by FTP Software, Inc. It is installed on many PCs on campus, an everything that has been said above about the NCSA package, applies to the PC/TCP package as well. The only difference: you should use the word `tn` in place of `telnet` on your command line.

Kermit Kermit is another public–domain software package. The old versions used to be only capable of making connections via a serial line, but now Kermit also has network support. To connect, assuming an IBM PC–compatible:

```
C:\> kermit
MS-Kermit> set line tcp/ip bohr
MS-Kermit> connect
```

Dial–in access You may access the Brock network by dialing in to our terminal server (688-4321) using a modem. This service requires that you set up a separate account with the Computing and Communication Services, in addition to your Unix host account. There is a charge for this service, currently $1 per hour of connect time. The details of how to establish a dial–in connection will depend on your home computer's communications software and you may need to look them up in the appropriate manuals. Eventually, a menu like this one should appear on your terminal screen:

```
      Welcome to Brock University's Dial-in Service
      ---------------------------------------------
   Enter a selection:

     * Server name (e.g. spartan) for connection
     * '?' or 'help' for help
     * 'exit' or 'quit' to terminate connection

   annex>
```

From here, simply type `bohr` to connect.

2.2 Logging in

Any one of the methods listed above should connect you to `bohr`, and you will immediately be asked for your login and password. When you connect to a Unix host, you should see a response from the system giving the system's name, the Unix version, and a login request. On `bohr`, the message looks like this:

```
Connected to bohr.physics.BrockU.CA.
Escape character is '^]'.

IRIX System V.4 (bohr)

login:
```

If you do not see the `login:` prompt within a few seconds, press the Return key again.

When the above login prompt appears, type your username in lowercase characters, followed by a Return. The next message you see will be a request that you enter the password. In response to this prompt, enter your password and press the Return key. This password will not be echoed, so that nobody can find out what it is by peeking at your terminal screen. Please note that Unix is case sensitive, so make sure to distinguish upper and lower case characters. Everything you type at a Unix command prompt must be in lower case unless otherwise specified.

Problems during login

1. After you connect to the Unix system via a dial–in connection, garbage is displayed on the screen instead of the `login:` prompt.

   Solution: check the parity and speed of the terminal. Set the parity to none, the number of data bits to 8, and the speed as appropriate for your modem. The Brock terminal server at 688-4321 currently supports rates of up to 14,400 bits per second.

2. After entering your login id and password, you receive the message `Login incorrect`, and the `login:` prompt reappears.

   Solution: Reenter your username and password. Keep in mind that Unix is case sensitive. If the problem persists, check the parity (see above).

> **Note**: If you have not logged in within one month of receiving your account, it will be disabled (for security reasons).

If, after several attempts, you are still unable to login, and you are sure you are doing things correctly, go see the person on duty at the Help Desk (F314) or call Brock extension HELP (4357).

Once you have successfully logged in, you should see something like this on your screen:

```
login: jdoe
Password:

IRIX Release 5.3 IP20 bohr
Copyright 1987-1994 Silicon Graphics, Inc. All Rights Reserved.
Last login: Fri Jul 14 10:44:34 EDT 1995
     You are now logged on as {jdoe}
bohr 1 %
```

and the `%` sign at the end of the last line above is the shell prompt that tells you that `bohr` awaits your command.

2.3 Logging in via Xterminals

Xterminals is the name of a relatively new class of terminal devices that use a very special way of dividing the work between the central Unix computer (server) and the desktop device from which a network connection originates. Telnet connections require very little sophistication from the desktop device, and all of the work is being done by the central server. An Xterminal–style connection requires much more from the calling desktop device, namely the ability to *render graphics*. The central server still does all the computing *per se*, but the presentation — managing multiple windows, keeping track of the mouse movements, static and animated graphics, shading and rendering of surfaces, *etc.* — is being done by the processor in your desktop device.

Hardware Xterminals are made by several manufacturers, but a high–end PC or Mac may have Xterminal–emulating software installed to make them perform just like a true Xterminal. The performance is typically a little slower, but otherwise this combination presents essentially the same interface and capabilities as the true hardware Xterminal does. It is also possible to run the Xterminal emulation software on a home computer, using dial-in access (see Section 2.1) and the so-called SLIP (Serial Line Internet Protocol) protocol, but the performance degrades significantly.

If you are on an Xterminal (such as the ones in B203), or once you have started the Xterminal emulator on your PC[1] (typically, this is done by double–clicking on an icon called `eXceedw` which looks like a big red X), you should see a list of hosts that are accessible from that particular location. Select `bohr` from that list (double–click on it, or select and then click on the OK button), and the login window should appear in the middle of your screen. The rest is very similar to logging in via telnet (see Section 2.2).

Once you are logged in, in the top left corner of your screen you should see a "Toolchest", a menu of several entries: Desktop, Selected, Find, System, Help. Feel free to browse through the Toolchest and try things out. The only word of caution is that since Xterminal protocol is still fairly new, some of the programs that you activate by clicking on the various buttons inside the Toolchest may not be Xterminal–compatible. They will simply refuse to run. For reference, the directories `/usr/bin/X11` and `/usr/local/bin` contain a lot of Xterminal–compatible programs, and their number is constantly increasing.

In addition to the Toolchest, there might also be a terminal window (probably labelled "xterm") on your screen. This is just like having a direct telnet session with `bohr`. The difference is, you can have many more sessions, opened all at once: simply activate the top entry under the Desktop menu in the Toolchest, the one called Unix Shell. Once you click on that button, a shadowy outline of a rectangular window will appear on your screen. Use your mouse to move the outline around; once you are happy with

---

[1] A special note for F317: look for an entry in the main blue menu that says "X terminal" and activate it.

its placement on your screen, simply click the left button, and you have opened another session! It will take some practice, but you should learn very quickly the purpose of the various buttons, bars, and pointers that you see on your screen. Most of them are quite intuitive. You will soon find that you probably have a particular way in which you prefer to organize your screen, with several windows displayed at once, doing different tasks in each one.

2.4 Changing password

If you are logging in for the first time, you probably want to change your password. To change your password, enter:

    % passwd

The `passwd` command will issue further prompts. When prompted for the

    Old password:

enter the password you have just used to log in. At the prompt

    New password:

type in your new password. You will then be asked to

    Re-enter new password:

Type in your new password again. This is to make sure both you and the computer agree on the password! If the password you typed the second time does not match the password you entered the first time, the system will not perform the password change, and you will be asked to try again.

A note about security

Your password is the key to the security of your Unix account. It is a good idea to change your password regularly. A good password is:

- at least six characters long
- a mixture of letters and numerals
- a mixture of upper- and lower-case letters
- not easily guessed (*i.e.* not any name, or your name spelled backwards, or any common word in any language, *etc.*)

When you change your password with the `passwd` command, the system performs some checking to verify that your new password is reasonably secure. It may reject the ones it deems too easy to guess, and you will have to try another one.

2.5 Logging out

To logout, just type

    % logout

at the shell prompt. If this does not work, try `exit`, or press Ctrl-d (press and hold Control key and then tap the d key).

Problems while logging out

1. If you get the error message:

       There are stopped jobs

   Solution: you have left some jobs running in the background, and cannot log out. You probably typed Ctrl-z somewhere along the way. Just type `logout` again: you will be logged out and the stopped jobs will be killed.

2. If you get the message:
   `logout: not a login shell`
   Solution: you are running a sub-shell. Type `exit` to exit the sub–shell, then try `logout` (repeat as necessary).

Logging out from Xterminals

On an Xterminal, you can use the above for each shell window that you have opened, but even if you do not have any windows left, the Toolchest is still there, so you have not yet logged out! The proper way to log out on an Xterminal is to either activate the very last entry (Log Out) under the System menu of the Toolchest, or to hold down the right mouse button anywhere on the background window. A menu containing a button labelled "Logout" should show up. Once you click on either one of these logout buttons, a separate dialog window will show up, asking you to verify whether you really want to log out. If you click on $\boxed{\text{Yes}}$, the system should close all remaining windows, log you out, and present the usual host selection list (see Section 2.3).

Chapter 3

Files and directories

No matter what your reason for using Unix, you are almost guaranteed to have to deal with files. This chapter describes files and directories and how they are organized.

3.1 File names

A file is simply a place where data is stored on a computer system. The contents of a file might consist of a program, the text of a paper you've written, or the data for a program. You can create files of your own or access files created by others. Once you create a file it is stored for you until you instruct the system to remove it.

Each file has a unique name. You can, within certain limitations, choose whatever names you want for files. Generally you can avoid trouble by creating filenames using only letters, numbers, periods (`.`), hyphens (`-`), and underscores (`_`). Many of the other characters have special meanings for Unix which might cause confusion in the interpretation of a name. It is also not a good idea to use spaces within file names. If you wish to use two words without having to run them together, separate them with a period, hyphen, or underscore, as in `chapter.one`. You'll probably aim for a balance, where filenames are long enough to be descriptive, but short enough to be typed easily and correctly.

3.2 Directories

Have you ever wanted to group files under common topics? A directory can be useful for doing just that. You may wish to have all programs relating to a certain project in one directory. That way, you know where all the programs and auxiliary files concerning that project are located. One of the best features of Unix is its use of directories. A directory is really no more than a special kind of a file. The difference is that it contains other files in a tree structure:

```
              /  ←—root
           ↙  ↓  ↘
        bin   etc   usr
       ↙↓↘   ↙↓↘   ↙↓↘
                       people
                       ↙↓↘
                adam  eve  mary
                 ↙     ↓     ↘
             temp    junk    fun
```

3.3 Changing Directories with `cd`

A graphical illustration of a directory tree structure is shown above. Suppose you start off in the root directory, `/`, (explained below). If you want to change directories to directory `usr`, you simply type

```
% cd usr
```

The **c**hange **d**irectory command is one of the most often used commands in Unix. Your home directory is in a directory tree. The root of the tree structure on Unix is the directory called `/`. If you execute the command

```
% cd /
```

then you will be at the highest point in the directory structure. You cannot `cd` to anything higher than the root directory.

In the preceding diagram, the user directories `adam`, `eve`, and `mary` would be situated in the tree structure as shown. The user `eve` has as her home directory the directory `/usr/people/eve`. The file `junk` has an absolute path name of `/usr/people/eve/junk`. The path to a file is the list of directories that lead to the specified file. If user `adam` wanted to look at the file `junk`, he could either

```
% cat /usr/people/eve/junk
```

or

```
% cat ../eve/junk
```

Note the `..` in the path name. The `..` is very special. It means the directory directly above the current directory. In other words, a

```
% cd ..
```

from the directory `/usr/people/eve` will put you in the directory `/usr/people`. Another special directory name is `.` (a single period). This refers to the current directory. Therefore, if you are in the directory `/usr/people`, you can look at the file `junk` by either

```
% cat ./eve/junk
```

or

```
% cat eve/junk
```

When you specify a path name without starting with a `/` as in this previous example, then the path name is said to be relative. If you use a `/` at the beginning of a path name, then the path is absolute, *i.e.* it starts at the root.

3.4 Printing the working directory with `pwd`

If you get lost and don't know where you are in the directory structure, you can always type

```
% pwd
```

This command will **p**rint your **w**orking **d**irectory.

3.5 Listing files with `ls`

You use the `ls` command to **l**i**s**t the names of the files in the current directory. It has many options, only a few of which are described here.

`ls -F` displays the files in the directory with certain indicators. Look at the following sample list of files:

```
% ls -F
a.out*   foo/    etc     goodies
sam.c    zut     zut.f
```

In this example the `/` indicates that `foo` is a directory. The `*` on the end of `a.out*` indicates that `a.out` is an executable file.

If you type `ls etc`, the file name `etc` is printed if that file exists. Typing `ls -A` lists all the files in the current directory. (By default, the "invisible" files with names starting with a `.` are not listed.)

`ls -l` shows the access permissions associated with the file. This option is described in more detail in Chapter 4.

3.6 Making directories with `mkdir`

Since a directory is just a special type of file, it follows the same rules for naming. To create a directory type

```
% mkdir name
```

and the directory `name` will be created in the current directory.

3.7 Removing directories with `rmdir`

The command to remove a directory is followed by the name of the directory you wish to remove. A directory must be empty before this will work. If any files remain in the directory, you will instead be reminded of their existence:

```
% rmdir foo
foo:  directory not empty
%
```

If you are sure you wish to remove the specified directory and all of its contents, then use

```
% rm -r foo
```

This will remove all files and subdirectories in the directory as well as the directory itself. Exercise care using this command, it has a great destructive potential.

3.8 Using wildcards

Unix file names can be of arbitrary length. To save yourself some typing, it is a good idea to learn about wildcarding. There are two characters that have very special meanings. The character `*` will expand to any set of characters. The character `?` will expand to a single arbitrary character. Their use is as follows. If you type

```
% ls f*
```

the names of all files starting with `f` will be displayed. If you type

```
% ls f?t
```

the names of all files with names exactly three characters long, starting with `f` and ending with `t` are printed.

Chapter 4

File protection

By default, a file that you create (including one created with an editor such as `vi`*), normally has permission for no other users to read that file (i.e. to scan its contents). Protecting your files from unauthorized perusal or making files readable by others can be done in a fairly straightforward manner. First, though, you must understand how to determine a file's protection mode.*

4.1 Determining a file's protection mode

Associated with each file is a set of protection bits. These bits determine whether somebody can read (`r`), write to (`w`), or execute (`x`) the file. There are three sets of bits for each file. The first set is for the owner of the file; the second set is for the group that the owner belongs to; and the third set is for other (everyone else). These sets of bits are best represented in the form of nine characters. If you type

```
% ls -l
```

certain statistics are printed about the files in the current directory. Here's an example for user `jdoe` who belongs to group `user`:

```
% ls -lF
-rw-r--r-- 1 jdoe user 38373 Jul  5 09:29 data
-rwxr-x--- 1 jdoe user 34756 Dec  4  1990 fly*
-rw-r----- 1 jdoe user 38373 Jul  5 09:29 fly.c
drwxr-x--- 2 jdoe user   512 Apr 23 13:09 foo/
```

On the first line, the first thing you notice is `-rw-r--r--`. Represented here are the protection bits (or mode) of the file. The first `-` indicates that it is not a special file (*e.g.* a directory). The next three characters are for the owner of the file (the owner can read [`r`] and write [`w`] the file, in this example). The three after that are for the group that the file belongs to (the members of the group `user` can read the file). The last three characters are for others (everyone else can read the file). On the last line, `foo` is indicated as a directory by the first character of the protection bits being a `d`.

Note, that since we used the `-F` switch in addition to the `-l` switch on the command line, the fact that `foo` is a directory is also indicated by the `/` at the end of its name.

4.2 Setting protection modes with `chmod`

The `chmod` command is used to alter the accessibility of your files for each of the types of users: owner, group, and other. The format of this command is

```
% chmod who[operation][permission] filename
```

where

$$\texttt{who} = \begin{cases} \texttt{u} & \text{for user (owner of the file } \texttt{filename}\text{)} \\ \texttt{g} & \text{for group (to whom the owner belongs)} \\ \texttt{o} & \text{for other (all others)} \\ \texttt{a} & \text{for all of the above} \end{cases}$$

$$\texttt{operation} = \begin{cases} + & \text{add the permission} \\ - & \text{remove the permission} \\ = & \text{set the permission (resetting all other permissions for } \texttt{who}\text{)} \end{cases}$$

$$\texttt{permission} = \begin{cases} \texttt{r} & \text{read} \\ \texttt{w} & \text{write} \\ \texttt{x} & \text{execute} \\ & \text{none}^1 \end{cases}$$

Examples

To remove all permissions for the group and all other users, so only the owner has access to the file `foo`:

```
% chmod go= foo
```

To give everyone read access to `foo`:

```
% chmod a+r foo
```

To turn a text file `bar` containing a Unix script into an executable file:

```
% chmod +x bar
```

Alternative form of `chmod`

This command can also be given in the form

```
% chmod mode filename
```

The `mode` is given as a three–digit number where the first digit controls owner access, the second controls group access and the third controls other access. The number is constructed by adding together the desired permissions according to the following table:

owner			group			others		
r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1

This is most easily seen by example. To give `rwx` access to the owner of `myfile` and none to group or others you would select

first digit (owner)	$4 + 2 + 1 = 7$
second digit (group)	0
third digit (others)	0

so the desired mode is 700 and the command to set it would be

```
% chmod 700 myfile
```

[1] A blank in the permission field, *i.e.* omitting any specific permission implies disallowing the access.

Chapter 5

Editing

This chapter discusses editing. It describes some basic commands in two text editors, `vi` *and* `pico`*. It is intended merely as a very quick introduction to the editors. Much more complete documentation is available for each editor, see Chapter 11 for details.*

5.1 Basic `vi`

Getting started

To edit a file with `vi` issue the command

```
% vi filename
```

The screen should then look like the figure below, with □ indicating the position of your cursor on the screen:

```
□
~
~
~
~
~
~
~
~
~
~
~
~
filename [new file]
```

Modes of operation

`vi` has two modes of operation: command mode and insert mode. You need to be careful to remember what mode you are in. If you are in command mode and press `x` you will delete a character. In insert mode, if you press `x`, a character "x" will appear in the document. When you first start `vi` you are in command mode. If you accidentally press some key that puts you into insert mode, pressing Esc should put you back into command mode. Note that as you perform this action, your cursor will move to the left one space.

Entering material

To begin typing in a document press `i` to put `vi` into insert mode. After that anything you type will be put into the document. You can type lines of text, pressing Return at the end of a line. When you have finished inserting text press Esc to get back into command mode.

Moving around

You can generally use the arrow keys on your keyboard to move the cursor around the screen. If the arrow keys do not work try pressing:

`k` for up

`h` for left `l` for right

`j` for down

Note that you have to be in command mode for these actions to work. If you get characters appearing in your document it is because you are still in insert mode and should press Esc to get back to command mode.

Deleting material

For all of the following operations you should first make sure you are in command mode, by pressing Esc.

Deleting a character To delete a character move the cursor over the character you wish to delete and press `x`. The character will disappear and the remainder of the line will shift to the left.

Deleting a word To delete a word, move the cursor to the first character of the word you wish to delete and press `d` `w` (two keystrokes, not echoed). The word will disappear and the remainder of the line will shift to the left.

Deleting a line To delete a line, place the cursor anywhere on the line you wish to delete and press `d` twice (*i.e.*, `dd`). The entire line will be deleted and the cursor will be placed at the start of the next line. To delete from the cursor position to the end of the line press `D` (uppercase).

Replacing material

Once again, for all of the following operations you should first make sure you are in command mode (by pressing Esc).

Replacing a character To replace a character move the cursor over the character you wish to delete and press `r` and then the replacement character.

Replacing a word To replace a word move the cursor to the first character of the word you wish to replace and press `c` `w`. The last letter of the word will be replaced with a $. You can now type the replacement word and then press Esc to get back to command mode.

Inserting material

Inserting text To insert text within a line, position the cursor to where the new text should go and press `i`. Enter the text and press Esc to get back to command mode. Note that the text is inserted before the cursor. To append text to the end of a line position the cursor at the last character of the line and press `a`.

Inserting a blank line Pressing o (lower case) inserts a blank line below the current line, while O (upper case) inserts a blank line above the current line. In both cases the cursor is placed at the start of the new line and `vi` switches to insert mode.

Saving your work

To save your work make sure you are in command mode and then enter `:w` followed by Return — a total of three keystrokes. The changes will be saved. If you type `:w newname` the document will be saved as `newname`.

Leaving the editor

If you want to save and exit type `:wq`. If you want to exit but not save your changes enter `:q!`. `ZZ` will write the file if changed and then exit.

`vi` is a powerful editor with many more commands and some very advanced ways of using them; this brief description barely scratched the surface, so to speak. Appendix B contains a slightly more expanded set of `vi` commands, and even more information can be obtained from the man pages. If you want to learn how to use `vi` efficiently you may want to consult some good reference book, see Chapter 11.

5.2 Basic `pico`

Getting started

To edit a file with `pico` use the command

```
% pico filename
```

The screen should look something like the figure below.

```
  PICO 1.7                                  File: filename
□

^G Get Help ^O WriteOut ^R Read File ^Y Prev Pg ^K Del Line  ^C Cur Pos
^X Exit     ^J Justify  ^W Where is  ^V Next Pg ^U UnDelLine ^T To Spell
```

At the bottom of your screen you will see a menu of commands that are available to you, and the notation ^ stands for Ctrl–. For example, use Ctrl-x to exit `pico`, Ctrl-o to write out (save) the results of the editing, *etc.*

Entering material

Unlike `vi`, `pico` is ready for you to type as soon as you enter it. Simply begin typing material into your document. As you reach the end of the current line, continue typing and the text will wrap onto the next line.

Moving around

You can generally use the arrow keys on your terminal to move the cursor around the screen. If the cursor keys do not work, try pressing

`Ctrl-p` (**p**revious line) for up

`Ctrl-b` (**b**ack) for left `Ctrl-f` (**f**orward) for right

`Ctrl-n` (**n**ext line) for down

If you try to move above the top or below the bottom of the screen, `pico` will scroll up or down to keep up with your cursor, provided that there are lines there. In addition, you can move up or down by a screenful, using `Ctrl-y` and `Ctrl-v`.

Deleting material

To delete a character move the cursor over the character you wish to delete and press Delete. The character will disappear and the remainder of the line will shift to the left. Alternatively, you can place the cursor to the right of the character you wish to delete and press Backspace.

To delete a line press `Ctrl-k`. This always deletes the entire line under the cursor.

Inserting material

To insert characters into existing text, position the cursor where you want the characters to go and type them; the cursor moves forward, and any text following the cursor moves forward as well. When you enter a file, `pico` is automatically in insert mode.

Justification

As you continue to type, `pico` will wrap lines to form paragraphs. One or more blank lines separate what `pico` considers paragraphs. To form blank lines, simply press Return. You can join two paragraphs together by deleting these blank lines using Delete or Backspace keys.

If after a long editing session a paragraph has many short and broken lines, place your cursor anywhere inside that paragraph and press `Ctrl-j`. This will re-justify the text to the margins of the screen.

Saving and leaving the editor

To save the changes you have made use `Ctrl-o`. This will write **o**ut the new version of the file, but will leave you in `pico` for further editing.

To leave `pico` press `Ctrl-x`. If you have made changes to your document since the last write out command, `pico` will ask if you want to save them and will verify the name of the file you want written out. If you specify a different filename at this point, the new file will be created containing the modified version, and the old one will remain unmodified.

Chapter 6

Some basic commands

The commands described in this chapter give you the basic file manipulating tools needed to get you started. These are by far the most basic of Unix commands and can be found on almost any version of Unix. More detailed information is available on-line for each command listed.

6.1 Copying a file with `cp`

Synopsis

`cp` *filename1 filename2*
`cp -r` *directory1 directory2*
`cp [-r]` *filename ... directory*

The `cp` command is used to **cop**y files. Its function is to copy the file name *filename1* to the file *filename2*.

Examples

```
% cp foo goo
```

results in the file `foo` being copied to `goo`. If the file `goo` exists and is writable by you, then the file `goo` is overwritten without warning. If the original file is a directory, the `-r` flag may be used to recursively copy all the contents of the directory.

```
% cp -r foodir /tmp
```

copies `foodir` and all of its files and subdirectories to the directory `/tmp`. This will create a subdirectory in `/tmp` called `foodir`, *i.e.* `/tmp/foodir`.

6.2 Moving or renaming files with `mv`

Synopsis

`mv [-fi]` *filename1 filename2*
`mv [-fi]` *directory1 directory2*
`mv [-fi]` *filename ... directory*

Use the `-i` option (for **i**nteractive mode) if you want to be prompted for confirmation if a move would replace an existing file or directory. Use the `-f` option to **f**orce a move. That is, it tries to change the file name even if the mode of the file looks like it might not allow this.

If the destination file is a directory, then the file(s) is (are) moved to the specified directory.

Example

To change the name of file `foo` to `goo`, type

```
% mv foo goo
```

6.3 Removing files with `rm`

Synopsis

`rm [-fir]` *filename ...*
`rmdir` *directory ...*

The `rm` command is used to **rem**ove files. Its purpose is to remove all files that are named.

Examples

```
% rm foo goo gum
```

removes files `foo, goo`, and `gum`. If the named file is a directory, then you need to specify the `-r` option to recursively remove all files in the directory and then the directory itself:

```
% rm -r foodir
```

removes the directory `foodir` and any files or subdirectories found in `foodir`.

Warning: Once a file is removed, it is gone forever and is no longer on the system. Use caution, especially with wildcards.

6.4 Displaying files with `cat`

Synopsis

`cat` *filename* ...

To look at the contents of a file you use `cat`. For example, enter

```
% cat foo
```

and the file `foo` scrolls right past you on your terminal at lightning speed. If you specify more than one file (*e.g.*, `cat foo goo`), both files are shown to you in the order you specified them. A friendlier file displayer is `more` (see description following).

A feature of `cat` you may find interesting is con**cat**enation (this is where `cat` gets its name). If you try

```
% cat foo goo > soo
```

then the files `foo` and `goo` are concatenated (put together) into the file `soo`. The `>` tells the shell to redirect standard output from your terminal's screen to the file `soo`.

6.5 Displaying files with `more`

Synopsis

`% more` *filename* ...

`more` also displays the named file(s) on the terminal's screen, but unlike `cat` it does it one screenful at a time. If the file is longer (in number of lines) than the number of lines on the screen, `more` displays `--More--(2%)` on the last line on the screen. The `(2%)` is the percentage of the file you have seen so far. If you press Space bar, the next screenful is displayed, and the percentage indicator increases. If you press Return, the text advances by a single line. At any time, you can press `q` to quit.

6.6 Listing processes with `ps`

Synopsis

`% ps [-f] [-u` *username*`]`

The command `ps` is used to show processes that are running on the system. A process is a job that Unix has been assigned to do, *e.g.* a program, a shell, a command, or a series of commands. To see all of your current processes, type

```
% ps
```

This displays something like

```
 PID TTY       TIME COMMAND
8853 ttyq4     0:28 -csh
9465 ttyq4     0:00 ps
```

The first field shows you the **P**rocess **ID**entification (PID) or process number. The second field shows you the device or port that the process is running on. In this case it is `ttyq4` for all of the processes shown (`TTY` is derived from the word teletype; `ttyq4` represents a specific device). The third field is the amount of processor time that the process has used. The last field is the actual process name and its arguments.

There are several other variations of the `ps` command display. See the `man page` for more details (man pages are explained in Section 6.8). Typing

```
% ps -f
```

gives more information:

```
  UID   PID  PPID  C    STIME TTY       TIME COMMAND
 jdoe  8853  8361  0 13:05:17 ttyq4     0:34 -csh
 jdoe  9615  8361  4 16:31:31 ttyq4     0:00 ps -f
```

This includes ownner's ID (`UID`), processor utilization used in scheduling the process (`C`), and process start time (`STIME`).

6.7 Terminating a process with `kill`

Synopsis

`kill [`*option*`]` *pid* ...

The `kill` command is used to terminate a command. If the command is running in background, you may specify a percent sign (`%`) and a job number instead of a *pid* (process id). For print jobs, for example, you can obtain the job number with the `lpq` command, which is explained in Chapter 7.

You can obtain *pid* of a process through the `ps` command as described above, and the *option* setting determines the severity of the kill. In its simplest form, the command

`% kill -9` *pid*

kills *pid* 'with extreme prejudice'. If this does not kill the process, nothing will. On the opposite end of the scale,

`% kill` *pid*

or

`% kill -1` *pid*

kills *pid* 'politely'.

6.8 Getting on–line assistance with `man`

Synopsis

`% man` *command*

The `man` command is used to access on–line documentation (**man**ual pages) for Unix commands. The information presented can appear a bit cryptic at first because the manual pages generally contain all possible options for commands.

Here's an example of a very short man page, the one for the Unix command `whoami`:

```
whoami(1)                 Silicon Graphics                 whoami(1)

NAME
     whoami - display the effective current username

SYNOPSIS
     /usr/bin/whoami

DESCRIPTION
     whoami displays the login name corresponding to the current effective
     user ID. If you have used su to temporarily adopt another user, whoami
     will report the login name associated with that user ID. whoami gets its
     information from the geteuid and getpwuid library routines (see getuid
     and getpwent, respectively).

FILES
     /etc/passwd     username data base

SEE ALSO
     su(1), who(1), getuid(2), getpwent(3)

NOTES
     The whoami command provided in versions of IRIX prior to 5.0 was a
symbolic link to /usr/bin/id(1).
                                                                  Page 1
```

There are several different sections to each man page. The first section, `NAME`, tells you the name of the command and a brief statement as to its purpose. The next section, `SYNOPSIS`, shows the full absolute path to the executable file and how the command is used. It also includes all of its options. The `DESCRIPTION` section tells you in more detail the purpose of the program (or command), and its uses. All of the `FILES` that are affected or used by the command are listed. Many man pages have sections for `OPTIONS` which includes a description of each of the options, and for `EXAMPLES` of how the command or program is used. The final section, `SEE ALSO`, tells you what other manuals to look at that are related to the man page you're viewing. There is also sometimes a section listing known `BUGS` in the program.

Some of the references in the manual pages are of the form `cp(1)`. The character inside the parentheses indicates what section of the on–line reference pages, collectively called *The Man Pages*, that particular entry can be found in. In the above case, the manual entry on `cp` can be found in section 1 of the *The Man Pages*.

Chapter 7

Printing

This chapter describes how to obtain printed output of files. It describes how to use various printers and how to make use of some of the printing options provided by the printers and the printing software.

7.1 Available printers

There are many different printers available at Brock. Line printers are typically capable of printing text only, but they do it very fast. Typically, you use such a printer to get a long listing out, such as a fortran program, or the output of a SAS run. There is one central line printer, referred to as `citoh` on many Unix machines on campus which can print to it. It is physically located behind the closed doors of the computer room, and the operators periodically bring out the printed output and file it in the appropriate boxes outside F300. Look for your printout under the first letter of your last name.

The other common type of printers is PostScript. This is a name of a popular "page description language" understood by many laser printers. Almost every software package you will encounter has the capability to produce output in PostScript–compatible form.

Several PostScript printers are available to Unix users at Brock. The following are of particular interest to Physics students:

`lw-physics` the printer of choice, this is the Apple LaserWriter printer in B203. Capable of printing both PostScript and text-only files. Your output is available for pickup whenever B203 is open, which is most of the time. Please, limit yourself to course–related printing on this printer.

`bj-physics` this is the BubbleJet printer in B203, also capable of printing both PostScript and text-only files. Rules of use are similar to `lw-physics`.

`ps` this is the central printer of the University's Computing Services. It is located in F301 and your output is available for pickup during the normal working hours.

`psd` the same printer! — except when addressed by this name, the multipage output sent to it gets printed in **duplex** mode, *i.e.* on both sides of the page.

`lj-physics` this is the HP LaserJet printer in the Physics Office. Its use is restricted to faculty, but if you need to use its features such as 600dpi resolution or duplex printing you may print there by a prior arrangement with Departmental secretary. Please make sure that your print jobs do not interfere with her work. Locked after hours.

`f316` as the name suggests, this printer is located in F316, and is only convenient when you are next door, in F317. Since some evening classes use those rooms, there usually is a TA on duty to give you your output.

`Tek` Tektronics is a PostScript colour laser printer in User Services (F314). It is expensive ($1.50 per page, $2.00 for transparencies) and should be used by arrangement with the HELP desk.

Department of Physics charges a nominal fee, to cover the cost of paper and printing cartridges for `bj-physics` and `lw-physics`. All other black-and-white printed output at Brock costs $0.10 per page, which is collected when you pick it up.

Setting your default printer

If you regularly use a certain printer you can set the default to that printer using the command `setenv`, described in Section 8.2. Here's an example:

```
% setenv PRINTER lw-physics
```

7.2 The `lpr` command

The Unix command to print a file is `lpr`. The syntax is

% lpr -P*printer filename*

where *printer* is one of the printer names described in the previous section, and *filename* is the name of the file you want to print.

The file must be in a format appropriate for the printer you are planning to use: text files for a line printer, or PostScript files for a PostScript printer. In general, it is your responsibility to make sure you are sending the right format of the file, although there is some error checking being done by the individual printers and by the `lpr` command itself. There are situations, of course, when the computer will do exactly what you ask it to do, and not what you had in mind. One example of that is sending a PostScript file to a line printer. PostScript is a set of readable — and printable — commands, but the line printer will not understand them, it will simply print them out as requested. The result is page after page of codes that look like complete nonsense.

There is no way to fix this up: only a PostScript printer is capable of understanding and printing PostScript files. The reverse, sending a text file to a PostScript printer, is much easier to achieve: a Unix program `lptops` does exactly that. You should consult the man pages to find out how to use this command. Note that on certain printers the text file will be correctly recognized as such and converted to PostScript as needed.

Examples

To print a PostScript file `foo.ps` on a PostScript printer `psd`, type

```
% lpr -Ppsd foo.ps
```

To print a text file `foo.txt` on the central line printer,

```
% lpr -Pcitoh foo.txt
```

To print the same file on the laser printer `ps`, use `lptops` to convert it to PostScript and "pipe" (send) the result to `lpr`:

```
% lptops foo.txt | lpr -Pps
```

To print more than one copy of your output, include `-#`*nn* on the `lpr` command line, where *nn* is the number of copies you want. For example,

```
% lpr -Pbj-physics -#3 foo.ps
```

would print 3 copies of `foo.ps` on the BubbleJet printer in B203.

7.3 Displaying printer queues

To see what files are in the printer queue (waiting to be printed), use `lpq` (**l**ist **p**rinter **q**ueue). Typing `lpq` without any options will print a listing of the printer queue for the default printer. If nothing is displayed then the queue is empty. Other printers are addressed in the same manner as for the `lpr` command. Typing

`% lpq -Ppsd`

will display the queue for printer `psd`.

7.4 Removing files from the queue

To remove files from the printer queue, (*i.e.*, to kill a print job) use `lprm`. Without any options, `lprm` will remove the current job from the default printer queue, provided that you own the job. If you use

`% lprm -Plw-physics -`

then all jobs you own queued for printer `lw-physics` will be removed from the queue. You may also specify a job number to remove jobs from the queue. The syntax would be

`% lprm -Plw-physics` *nn*

where `lw-physics` is the printer which may soon be printing the job and *nn* is the job number obtained from `lpq`.

Chapter 8

The C shell

As mentioned previously a Unix shell *is a special kind of program. It takes commands from the keyboard, processes the command line, and then runs the command given on the command line. In this chapter we will discuss one particular shell, called the* C Shell *(*`csh`*). This is the default shell that is specified when accounts are set up by the CCS.*

8.1 Shell start-up

Whenever a `csh` is started, it automatically searches for and runs commands in several key files. These files are located in your home directory, *i.e.* the directory in which you are placed each time you log in.

The `.cshrc` file

Every time the `csh` is started, it searches for a file in your home directory called `.cshrc`. This file can contain any Unix commands as well as any of the built-in commands of `csh`. A sample `.cshrc` file is shown in Figure 8.1. Note that all lines beginning with `#` are comment lines and are ignored by `csh`.

The `.login` file

Whenever you log in, after `.cshrc` is executed, `csh` also executes a file in your home directory called `.login`. This file follows the same rules for contents as the `.cshrc` file. The only difference is that this file is only executed once, when you first log in, rather than every time you start a command shell. A sample `.login` file is shown in Figure 8.2.

8.2 Shell commands used in `.cshrc` and `.login`

Fig. 8.1 and 8.2 of the previous Section show how certain shell commands can be executed at startup by placing them inside shell startup files `.cshrc` and `.login`. These commands can also be executed interactively, from the shell prompt. Here's a brief discussion of the commands you are likely to need to begin personalizing your computing environment.

The `history` command

The `history` command provides a mechanism for "remembering" a list of recently used command lines and for re-entering those commands. To use this feature you must first issue the command

`% set history=`*nn*

where *nn* is the number of commands you wish to remember. This is set to 50 in the default `.cshrc` file. You can reset it to a different value if you wish. Once it has been set you can issue the command

```
#       .cshrc

#..  default mask on file permissions for the files I create
umask 027

#    skip remaining setup if not an interactive shell
if ($?USER == 0 || $?prompt == 0) exit

# prompt -- set the system prompt
set prompt="`hostname`% "
#    When something is in between ` and ` the C shell (csh) takes that
#    to mean "run the enclosed command".  In this case, hostname is run
#    to have the prompt show the name of the current host.

# history -- set the number of commands to remember
set history=50

# alias -- make an alias for a command
#    syntax:  alias new_command old_command
alias cd   'cd \!*; ls -FC'
alias h    history
alias la   ls -la
alias ls   ls -FC
alias lib  telnet nexus 1200
alias bye  logout
alias del  rm -i
alias x    chmod +x
```

Figure 8.1: A sample `.cshrc` file.

`% history` *n*

where *n* is a number less than *nn*. This will display the last `n` commands. If you omit the *n*, the full set of *nn* commands are displayed. You can subsequently enter

`% !`*n*

to re-issue the *n*-th command from the displayed list, or

`% !`*str*

to re-issue the last command in the history list that starts with the string *str*.

The `alias` command

The alias command allows you to define new commands or to make up your own abbreviations for frequently used commands. The form of this command is

`% alias` *new command*

where *new* is a new command or abbreviation and *command* is the command you wish to have executed when you type *new*. For example

`% alias delete rm -i`

would define `delete` as an alias for the command `rm`. Note that you can use `alias` to re-define existing commands but this should be done with caution. If you type `alias` by itself, csh will display a list of current aliases.

You can remove an alias with the command

`% unalias` *name*

```
#       .login file

# Read in after the .cshrc file when you log in, and not for subsequent
# shells.  For setting up terminal and global environment characteristics.

# path -- The directories to search for commands that can be run
set path=(.  /usr/bin /bin /etc /usr/local/bin /usr/sbin /usr/bsd /usr/bin/X11)

# set the terminal type
if ( -d /usr/lib/terminfo) then
   set ignoreeof noglob; eval `tset -s -Q`; unset noglob
endif

# Set the default X server
if ($?DISPLAY == 0) then
   if ($?REMOTEHOST) then
      setenv DISPLAY ${REMOTEHOST}:0
   else
      setenv DISPLAY :0
   endif
endif

# environment variables
setenv EDITOR                  /usr/bin/vi
setenv TRIUMF_FONTS          /usr/local/lib/triumf
setenv TRIUMF_TERMINAL_TYPE X
setenv TRIUMF_PLOTTER_TYPE  POSTSCRIPT
setenv PRINTER                 lw-physics
```

Figure 8.2: A sample `.login` file.

where *name* is the alias you wish to remove.

Aliases are usually defined in your `.cshrc` file so that they are available each time you log on.

The `setenv` command

The `setenv` command is used to set or change various aspects of your working environment. It can be used, for example, to change various defaults, such as the default printer or the type of terminal. The general format is

`% setenv` *VAR word*

where *VAR* is the environment variable that you wish to set and *word* is the value to set it to. For example, to set your default printer to `bj-physics` the command would be

`% setenv PRINTER bj-physics`

To set the terminal type to vt100 the command would be

`% setenv TERM vt100`

To set your editor of choice to `pico`

`% setenv EDITOR pico`

You can make these settings permanent by placing them in your `.login` file, (as shown in the sample file of Section 8.1) using `pico` or `vi`.

8.3 Control characters and special keys

Control characters are typed by holding down the Control key and simultaneously typing a character on the keyboard, while *special keys* perform a function by themselves. Both can be extremely useful in saving time and effort. As a rule, a control character or a special key will not always perform the same functions for a particular program as it does for the shell; thus you are advised, as always, to be aware of the consequences of what you are doing. The functions below apply specifically to the shell, though they may be applicable elsewhere, such as with some Unix utility programs.

Moving the cursor

The functions in this section accomplish rudimentary cursor movement, both horizontally and vertically. Though they are described with a video display screen in mind, they will perform similarly on a hard copy terminal (a computer without a monitor that uses printed output for its display).

Ctrl-i or Tab – moving over eight spaces. This function is performed by both a control character and a special key. Default tabs are usually set at eight character intervals (the intervals may be adjusted on the terminal). The function default is to move the cursor over to the next column of eight without echoing anything on the terminal. This function is commonly used by programmers for indenting and making readable code.

Return – terminating a command. This key (labeled Enter on some keyboards) is usually used as a terminator for a command line; it indicates to the shell that the command is complete. The key is often referred to as a *carriage return* (CR for short) since it returns the cursor (as opposed to the carriage of a typewriter) to the left margin.

Controlling terminal display

Two control characters can be used alternately to pause (or suspend) and resume terminal output or display:

Ctrl-s – suspending output. Ctrl-s temporarily halts the scrolling of data as it is being displayed on the terminal. This function is fairly unnecessary if the terminal is set to halt after each page, as it would with a pager such as `more`.

Ctrl-q – resuming suspended output. Ctrl-q is used to resume output after having typed a Ctrl-s to suspend output. If a terminal is "not working", try typing Ctrl-q. Often you've typed Ctrl-s unknowingly.

Editing the command line

Several options are provided to correct errors made while typing a command line to the shell: you have the choice of deleting single characters, words, or deleting the entire line. The effects of the functions given in this section are identical for all terminals, but the manner in which the terminal shows the execution of the function depends on the terminal parameters.

Delete or Backspace – delete character. Either one of these keys can be set up to delete the previous character and move the cursor back one space.

Ctrl-w – deleting a word. Ctrl-w is useful for deleting single words in command lines; it acts on the word immediately to the left of the cursor. A *word* is considered to be a string of characters preceded by a blank or the beginning of the line. The word will either be erased or the line will be redrawn.

$\boxed{\textbf{Ctrl-u}}$ – **delete an entire line**. $\boxed{\text{Ctrl-u}}$ is used to delete the current command line. If you type a command, then type CTRL-u, the system will not process it. $\boxed{\text{Ctrl-u}}$ either erases the current line or starts on a new line.

Job control

$\boxed{\textbf{Ctrl-c}}$ – **halting command execution**. $\boxed{\text{Ctrl-c}}$ will interrupt execution of a command or program that is currently running. It will also delete the current command line if typed when entering a command.

$\boxed{\textbf{Ctrl-z}}$ – **suspending execution**. Typing $\boxed{\text{Ctrl-z}}$ will cause the currently executing program to be suspended. This means the program will hang in limbo until either it is "killed" or is resumed. To resume execution type `fg` (`fg` stands for `f`oreground). To resume execution but run the program in the `b`ackground, type `bg`. To terminate the program, see the section on `kill` in Chapter 6.

Chapter 9

Communicating with others

Brock's Unix servers provide access to electronic mail, or e–mail. They are directly connected to the Internet, and make use of the campus e–mail gateway for relaying messages to and from other mail networks. This chapter discusses mail in general, gives a brief introduction to three mail programs, and discusses ways of finding out information about others on the system and on other systems.

9.1 Electronic mail (e–mail) in brief

Mail programs

If you use standard Unix mail, you have a choice of several user interfaces; the standard Unix `mail` (AT&T style) or `Mail` (Berkeley style) commands, and the public–domain mail programs `elm` and `pine`.

Unix servers also provide a *POP server*, which allows you to use `Eudora` on a Macintosh and `NUPop` on a PC to send and access mail through your Unix account.

Receiving mail

If you receive the message

```
You have mail.
```

or

```
You have new mail
```

when you first log in, there is mail waiting in your mailbox. You can use one of the mail programs described below to read your mail.

Sending mail

Sending mail to a person on the local machine requires knowing only their username. Sending mail to remote hosts requires slightly more information. Remote mail addresses take on different syntax depending upon the network on which the destination machine resides.

You can use the `finger` command, described later in this chapter, to find out a person's username.

9.2 The `mail` command

The standard Unix `mail` command provides a command line interface allowing you to read, compose, send, and print your messages. You start the mail program with the command

```
% mail
```

Reading mail with `mail`

Once `mail` has been started, it will show you a summary of the messages waiting and prompt you for a command. To read a message, you merely type the number of the message as shown in the summary at the `mail` prompt (usually, a question mark, `?`). `mail` then displays the message.

Once you have read the message, type

```
? delete
```

and the message will be deleted. Or, if you want to save the message, type

```
? save filename
```

and the message will be saved in `filename`. After saving the message, you might want to delete the message with the `delete` command or else the message will still be in your mailbox. (A mailbox is a special file where mail is stored). By default, exiting normally from `mail` will cause any undeleted messages to be moved to a file called `mbox` in your home directory.

To exit `mail`, simply type

```
? quit
```

Sending mail with `mail`

To send mail to another user on the same system using `mail`, just type

```
% mail username
```

At this point, when you are asked for

```
Subject:
```

enter an appropriate subject for this message. You are then free to start typing in your message. If you would like to edit the message using an editor, hit Return followed by `v` and another Return. This will place you in the editor. After exiting from the editor or after you have finished typing in the message, you may send it by typing Ctrl-d or `.` – a single period. Both must be typed as the first character on a new line.

9.3 The `Mail` command

`Mail` (capital `M`) is a slightly more friendly version of `mail`. It allows you to create folders for your mail, aliases for frequently used mail addresses, *etc.*, but in general it is a close relative of `mail`.

9.4 The `elm` mail program

The `elm` mail program provides a full–screen interface that displays a menu of messages and allows you to select from the menu. It uses one of the full-screen Unix editors for composing outgoing messages.

To start `elm` simply issue the command

```
% elm
```

You will be presented with a screen which looks like the one in Figure 9.1.

Reading Mail with `elm`

You will note that the main screen presents you with a numbered list of your messages. The current message is either highlighted or indicated with an arrow pointing to it. To read that message you simply press the Return key. To select a different message use the arrow keys on your keyboard. (If the arrow keys do not work you can use the j and k keys to move up and down through the list.)

When you have read the message press `i` (for **i**ndex) to return to the main screen.

```
Mailbox is '/usr/spool/mail/jdoe' with 1 message [ELM 2.3 PL6]

-> 1  Aug 30 Shelley R. Watson  (31)    Re:  emacs editor
   2  Aug 31 Fred Wilson        (12)...Re:  animal crackers






You can use any of the following commands by pressing the first character;
d)elete or u)ndelete mail, m)ail a message, r)eply or f)orward mail, q)uit
To read a message, press <return>.  j = move down, k = move up, ?  = help

Command: □
```

Figure 9.1: A sample `elm` screen. □ indicates the cursor position.

Sending Mail with `elm`

To send a message press the $\boxed{\text{m}}$ (for **m**ail) key while you are at the main screen. You will be prompted for recipient, subject, and who to send copies to (prompted as `Cc:`), then the `vi` editor will be invoked for you to use to compose your message. (See Chapter 5 for instructions on using `vi`.) When you are finished, exit from `vi` and press $\boxed{\text{s}}$ to send the message.

To leave `elm` simply press the $\boxed{\text{q}}$ key while at the main screen.

9.5 The `pine` mail program

`pine` was developed at the University of Washington and is used on many Unix systems at Brock. `pine` is based on `elm` and thus to a large extent the functionality is very much as described in the previous Section, but the interface has been re-designed for the greatest ease–of–use.

You start `pine` with the command

```
% pine
```

The most valuable feature of `pine` is the set of prompts that are always displayed at the bottom of the screen listing the commands available to the user at any given time. `pine` comes with its own built-in editor `pico`, described in Section 5.2. Consult there for the example of the screen appearance, and for the descriptions of basic `pico` commands.

9.6 Forwarding your mail

If you would like your mail forwarded to another account on your current system, or to another account on another machine, then you need to create a file called `.forward` in your home directory (and make it readable by everyone). This file should consist of the address or addresses to which you want your mail forwarded. For example, if you want your mail forwarded to the account `joe` on the same (local) machine as the account from which you want your mail forwarded, the `.forward` file should contain the line

```
joe
```

If you want your mail forwarded to the account `sam` on the machine `pollux`, then your `.forward` file should contain the line

```
sam@pollux
```

More than one address may be specified as long as each address is separated by a comma or newline.

9.7 Talking with other users

If you wish to talk to another user right away, the command to use is `talk`. This command sets up a two–way interactive session on your terminal screen or window. It divides the screen into two equal halves: the top half is for the messages that you type and the bottom half for the messages that the other user types. The screen is updated each time a user types in a character. To start talk, type

```
% talk user
```

or

```
% talk user@remote-host
```

This will draw the screen on your terminal and send a message to the user with whom you are trying to communicate. The very top line of your screen will show you the status of your request.

If you receive a message similar to

```
Message from Talk_Daemon@chilko.usc.edu at 11:26 ...
talk:  connection requested by user@host
talk:  respond with:  talk user@host
```

then you are being asked by someone to `talk` with them. You may respond by using the last line:

```
% talk user@host
```

This will draw the screen and automatically connect you with the person who initiated the conversation. To end a `talk` session, just hit the interrupt key (usually Ctrl-c).

9.8 Obtaining information about others

Seeing who is on the system with `who` and `w`

The commands `who` and `w` can be used to show who is currently on the system. `who` will show you just who is on and when they originally logged in. `w` will show you who is on, when they logged in, what process they are currently running, and several other things.

```
% who
adobos     ttyq0        Sep  9 09:44
rael       ttyq3        Sep  9 08:23
srothste   ttyq13       Sep  9 09:59
kgough     ttyq15       Sep  9 10:02
% w
 10:04am  up 8 days,  8:45,  6 users,  load average: 4.84, 4.54, 4.22
User     tty from           login@  idle   JCPU   PCPU  what
adobos   q0  annex.ac.Brock 9:44am            1      1  telnet badger
rael     q3  hds2.physics.b 8:23am         2:42         sh -c gnuplot_x11
srothste q13 139.57.81.101  9:59am     4                /usr/bsd/more
kgough   q15 139.57.192.102 10:02am                     vi pentanegt.freq
```

Finding information about individuals with `finger`

You can use the `finger` command to find out information about other individuals on the system and on other systems. The format of the command is

% finger *name*

where *name* can be a user name or a person's first or last name.

```
% finger srothste
Login name: srothste                    In real life: Stuart Rothstein
Directory: /homes/fms/chemistry/srothste        Shell: /bin/csh
On since Sep  9 09:59:32 on ttyq13 from 139.57.81.101
23 seconds Idle Time
No Plan.
```

Chapter 10

Access to remote systems

There are several programs that allow access to remote systems. `telnet` *and* `rlogin` *allow you to log on to remote systems, and* `ftp` *provides a mechanism for transferring files from one system to another. To use these programs you need to know the name of the system or host you wish to connect to. In all cases, a login name and password are required on the remote host.*

10.1 Transferring files with `ftp`

`ftp` provides a fast method of transferring files from one system to another. You can start `ftp` with the command

`% ftp` *hostname*

where *hostname* is the computer you wish to connect to. The remote system will then prompt you for a login name and a password. Once the login is complete you should see the prompt

`ftp>`

At this point you can enter an `ftp` command. There are a number of commands, but only the most commonly used are given here.

`ftp> cd` *directory*

changes the working directory on the remote machine to *directory*.

`ftp> lcd` *directory*

same, but applies to the local machine. Allows you to transfer files to/from directories other than the one active at the time you started `ftp`.

`ftp> binary`

This sets up ftp to transfer binary files instead of text files (the default).

`ftp> get` *filename [filename2]*

Retrieve *filename* and store it on the local machine. If *filename2* is not specified, it is given the same name it has on the remote machine.

`ftp> put` *filename [filename2]*

Transfer a file on your machine to the remote machine. If *filename2* is left unspecified, the original name of the file you're transferring is used.

`ftp> pwd`

Print the name of the current working directory on the remote machine.

`ftp> close`

Terminate the `ftp` session with the remote server, and return to the command interpreter. At this point you can

`ftp> open` *hostname*

This opens a connection to another *hostname* without leaving `ftp`. This command also provides an alternative way of starting your `ftp` session in the first place:

`% ftp`
`ftp> open` *hostname*

Finally, the commands

`ftp> quit` *or*
`ftp> bye`

terminate the `ftp` session with the remote server and exit `ftp`.

For more information on `ftp` and a complete description of the commands see the appropriate man page (see Section 6.8).

Anonymous ftp

A number of places maintain collections of programs and data and allow access to these collections via "anonymous ftp". To get access to these systems you enter `ftp hostname` as described above. When prompted for a login name, enter `anonymous`. If prompted for a password enter your own name and system (`jdoe@bohr.physics.brocku.ca`, not echoed). Note that with anonymous `ftp` you can normally only get files from the remote system, not put them there.

10.2 Transferring files with `rcp`

The **r**emote **cop**y command, `rcp`, is very similar to the regular copy command (see Chapter 6), except it allows one or both of the source/destination files to be files residing on other computers. A typical use for `rcp` is to transfer a data file acquired on a PC during a mechanics lab up to the Unix host for processing and analysis, like this:

```
C:\> rcp exp5.dat jdoe@bohr.physics:p220/exp5.dat
```

Note the special characters: `@` delimits the user name to be used on the remote host, and `:` separates the host name from the file name on that host. In the above example, the local file `exp5.dat` is copied into a subdirectory `p220` of `jdoe`'s home directory on `bohr.physics`.

If you have accounts on two separate Unix hosts, `rcp` can be used to transfer data between them. Note that the use of wildcards is usually limited to local files only.

10.3 Logging on with `telnet`

To start `telnet` issue the command

`% telnet` *hostname*

or

`% telnet`

In the first case `telnet` will establish a link to *hostname* and you will be prompted for a login name and password. In the second case `telnet` will respond with the prompt

`telnet>`

and you may enter a `telnet` command. To open a connection issue the command

`telnet> open hostname`

Once a connection has been established all text that you type is sent to the remote host. When you log off the remote host, either the `telnet` session will be terminated or you will return to the `telnet>` prompt. In the latter case you can type

`telnet> quit`

to terminate `telnet`.

10.4 Logging on with `rlogin`

`rlogin` allows you to log on to a remote machine computer in a slightly different way. The main advantage of `rlogin` over `telnet` is that you can set up automatic username recognition on the remote machine, and then typing of the name and password will not be necessary. Consult `rlogin`'s man pages for further information.

Chapter 11

Further information

This chapter describes how to get further information on many of the topics in this manual. It includes descriptions of commercially available books, other CCS documentation, print-ready documents available on line, and getting printed copies of man pages.

11.1 Commercial books

A manual that CCS recommends as a general Unix reference book is *A Practical Guide to the Unix System*, second edition, by Mark G. Sobell, published by Benjamin/Cummings (1989).

Using the `vi` *editor*, by Susan Ng, University of BC (1991), is an excellent "intermediate" manual: much more detailed than the description presented in Chapter 5, but still more compact than most commercially available books on `vi`.

This document was prepared using a powerful page layout language called TEX with the LATEX package of standard macros. Both are available on Brock Unix servers. LATEX – *A Document Preparation System* by Leslie Lamport, published by Addison-Wesley Publishing is a very good book if you would like to write your own documents using LATEX.

O'Reilly & Associates, Inc. has a series of detailed books on specific Unix–related topics, including detailed manuals on some commands such as `vi`.

Some of these books are available in the library, or a short–term loan of the Departmental copy can be arranged with Frank Benko. You may also order your own copy through the Brock Bookstore.

11.2 Other relevant CCS documentation

CCS sends out a *Newsletter* on a regular basis which contains information that might be useful.

User Notes are brief documents prepared by the User Services of the CCS. The following have been printed to date:

- Using Berkeley Mail at Brock U.
- Quick reference to `vi`
- Moving Unix files to a Mac or PC using Kermit
- Introduction to the Macintosh
- Installing Eudora
- Using Telnet on the Macintosh
- Using ftp on the Macintosh
- DOS Introduction
- Using Telnet on the PC
- Using ftp on the PC

Other User Notes may have become available since this document was composed. The most up-to-date list of User Notes, copies of specific User Notes, and past issues of the *CCS Newsletter* are available through the **Help Desk** in F314 (ext. 4357 or e–mail to `helpdesk@spartan.ac.brocku.ca`).

11.3 Print-ready documents available on-line

User Services and the Unix Support Group of the Systems Services are currently transferring most of the available documentation into electronic form, to be available on-line. Some of it is already in place, and available via `gopher` and `Mosaic`. Please, consult the **Help Desk** for the status of this project.

11.4 Printing man pages

Virtually all Unix commands and utilities have man pages available for them. You can look at the pages on-line but all man pages can be printed out as well. To be properly formatted, man pages need to be printed on a PostScript–capable printer. To have a page printed using CCS's PostScript printer, issue the following commands:

```
% setenv TCAT lpr
% setenv PRINTER ps
% man -t page
```

where *page* is the man page (the topic) you wish to print.

Note that if you want to print multiple man pages in one session you need only give the first two lines once. Your output will be printed on the PostScript printer and will be available from the CCS Reception Area (F301).

Appendix A

Summary of Unix commands

Command	Description
`cd` *dir1*	change directory to *dir1*
`pwd`	print the working directory name
`more` *file*	display the contents of *file*
`less` *file*	display the contents of *file*
`pico` *file*	edit *file* using *pico*
`vi` *file*	edit *file* using *vi*
`ls`	list file names in the current directory
`cp` *file1 file2*	copy *file1* onto *file2*
`mkdir` *dir1*	create a new directory called `dir1`
`mv` *file1 dir1*	move *file1* into *dir1*
`mv` *file1 file2*	rename *file1* to *file2*
`rm` *file1*	delete *file1*
`rm -ri` *dir1*	delete *dir1* and any files it contains with confirmation
`man` *keyword*	display manual page for *keyword*
`apropos` *keyword*	locate commands by *keyword* lookup
`man -k` *keyword*	locate commands by *keyword* lookup
`lpr` *file*	print file to the line printer
`gopher`	information about Internet
`info`	same as gopher
`ftp` *site*	enable file transfers to and/or from *site*
`help`	get online help

Appendix B

Summary of vi commands

Command	Description
Starting `vi`	
`vi` *filename*	open or create file
`vi +18` *filename*	open file to line 18
`view` *filename*	open file read-only
Cursor Commands	
`h`	move left (west)
`j`	move down (south)
`k`	move up (north)
`l`	move right (east)
`w`	move right (ahead) one word
`b`	move left (back) one word
`^`	move to beginning of line
`$`	move to end of line
`H`	move to home position
`L`	move to lower left corner
`M`	move to middle of screen
`G`	move to end of file
`1G`	move to beginning of file
`21G`	move to line 21 in file
`Ctrl-u`	scroll up
`Ctrl-d`	scroll down
`Ctrl-f`	page forward
`Ctrl-b`	page back
Setting Line Numbers	
`:set nu`	show line numbers
`:set nonu`	hide line numbers
Saving and Quitting	
`:w`	save changes (write buffer)
`:wq`	save changes and quit vi
`ZZ`	save changes and quit vi
`:q!`	quit without saving changes

Command	Description
Inserting and Appending Text	
`a`	append after cursor
`A`	append at end of line
`i`	insert before cursor
`I`	insert at beginning of line
`o`	open line below cursor and enter insert mode
`O`	open line above cursor and enter insert mode
Changing Text	
`cw`	change word
nn`cw`	change *nn* number of words
`cc`	change line
`C`	change part of line to right of cursor
`r`	replace character under cursor
`s`	substitute string for character under cursor
`J`	join current line and line below
`xp`	transpose character at cursor and character to right
`~`	change case of letter (upper or lower)
`u`	undo previous command
`.`	do last change again
`:u`	undo previous last-line command
`r`	replace character under cursor
Deleting Text	
`x`	delete character under cursor
`X`	delete character before cursor
`dw`	delete word
`d`*nn*`w`	delete *nn* number of words
`dd`	delete line
nn`dd`	delete *nn* number of lines
Copying and Moving Text	
`yw`	yank word to copy
`yy`	yank line to copy
`y$`	yank to end of line
`dd`	delete line
`p`	put yanked or deleted line below current line
`P`	put yanked or deleted line above current line
Searching and Replacing Text	
`/`*string*`/`	search forward for *string*
`?`*string*`?`	search backward for *string*
`n`	find next (or previous) occurrence of *string*
`N`	repeat search in the opposite direction
`:1,$s/`*string1*`/`*string2*`/g`	search for every *string1* and replace with *string2*

Index