CS1000

An Introductory Manual to the Unix Operating System and the Computer Sciences Department's Instructional Computing Environment

Computer Systems Laboratory
Computer Sciences Department
University of Wisconsin, Madison

Spring 2008

Preface

The Computer Systems Laboratory distributes CS1000 to assist users new to the Unix environment at the University of Wisconsin.

CS1000 was first created by Brian Pinkerton and Tom Christiansen. It was later revised by Nichole Wolfe and endured many corrections from Mitali Libeck. Brent Halsey cannibalized it and converted it to LaTeX. Mike Bloy and Ryan Fruit started from scratch on the LaTeX source and cleaned it up. Becky Solomon, Nick Petska, Jacob Ela, the notorious Mick Beaver, and Peter Koczan all made drastic overhauls to bring the content out of the stone age.

Although much care has been taken putting this together, it is possible that some mistakes got through or have been overlooked. If you do find any mistakes, or would like to comment on this document, please send mail to *lab@cs.wisc.edu*. Your feedback is very much appreciated.

World Wide Web

This document is also available via the World Wide Web. The HTML version is at *http://www.cs.wisc.edu/csl/cs1000/* and the PDF can be downloaded from *http://www.cs.wisc.edu/csl/cs1000.pdf*. Other URLs in this document will point to root documents, rather than to specific documents within a site tree. This is because the Web is a mutable environment, and any links we put into a printed document may change as quickly as the day of printing.

Conventions

The following typographic conventions are used in this book:

- **Bold** is used to draw attention to machine names and other labels you will see. It is also used to highlight words for the first time.
- *Italics* are used to draw attention to filenames and command names. It is also used to highlight important notes and points.
- `Constant Width` is used for code and screen examples.
- *`Constant Width Slant`* is used for user input in examples. It is also used for electronic mail addresses and URLs.

References

By necessity, this document is relatively short and cannot be comprehensive. The following other sources of information are recommended:

Computer Systems Lab Information and Documentation - *`http://www.cs.wisc.edu/csl`*

The UNIX System Home Page - *`http://www.unix.org`*

Google - *`http://www.google.com`*

Learning the UNIX Operating System - Grace Todino, John Strang, and Jerry Peek. *O'Reilly & Associates, Inc.*

UNIX for Dummies - John R. Levine & Margaret Levine Young. *IDG Books.*

UNIX in a Nutshell - Daniel Gilly and staff. *O'Reilly & Associates, Inc.*

UNIX Power Tools - Jerry Peek, Tim O'Reilly, Mike Loukides and staff. *O'Reilly & Associates.*

Contents

Chapter 1

Introduction

Welcome! This document, CS1000, is a primer designed to help you adjust to the Computer Sciences Department's Unix computing environment. Most likely, you will also want to attend one of the Unix orientations given during the first two weeks of classes.

Unix™ is an operating system, just like MS-DOS™, Microsoft Windows™, or MacOS™. It runs on a wide range of computers. It is precisely this element of portability that has allowed Unix to become as widely used as it is today. It has risen above its humble beginnings as a programmer's toy on an old minicomputer to become the operating system of choice not only at universities and other research facilities, but also on computers ranging from huge mainframes to the microcomputers that professionals from many fields use. Additionally, most courses you take from this department will be taught using a computer that runs Unix of one flavor or another. For these reasons, familiarity with Unix will aid you both in later life as a computer professional, as well as with your assignments in this department.

Remember as you are reading and learning that nobody learns Unix in a day. This document is by no means comprehensive and barely scratches the surface of some of the more advanced topics and features in Unix. Fret not, gentle newbie, for Unix, like anything worthwhile, takes time to learn and even the most experienced users and hackers have gaps in their knowledge. Be patient, have fun, and seek out help and answers where you can.

NOTE: References to the "demo01.cs.wisc.edu" computer or various computer labs should be viewed only as examples. In the latter case, there may or may not be a lab with computers of that name, as the lab names often change and get recycled.

1.1 About This Document

- Chapter 1 is this Introduction. You're reading it right now.
- Chapter 2 of this document will lead you through logging on for the first time.
- Chapter 3, by far the longest chapter, will cover the essential things about the interface that you need to know.
- Chapter 4 covers the various ways of getting help from the system.
- Chapter 5 covers some advanced topics that may come in handy.
- Chapter 6 is a Quick Reference to commands covered elsewhere in the document.

1.2 About Your Account

You are reading this document because you have an account on the UW-Madison Computer Sciences Department's instructional Unix workstations. This account gives you access to email, a place to put a web page, disk space, and access to information. This account is seperate from your DoIT WiscWorld account. We hope that the experience will prove both enlightening and fun.

1.3 The Systems Lab

The Computer Systems Lab, known as the **CSL**, maintains and operates a wide variety of systems for instructional and research use for the Computer Sciences Department at the UW-Madison. Its staff of undergraduate and graduate students and full-time professionals performs file backups and restorations, installs new systems, corrects bugs in existing software, and develops new software. The CSL should never be used as a substitute for your TA, instructor, or a Unix consultant stationed in the user rooms. Only matters pertaining to the functioning, administration, or maintenance of system hardware or software should be brought to the attention of the CSL. The CSL, located in room 2350, is open Monday through Friday from 8:00am to noon and 1:00pm to 5:00pm, with limited evening and weekend coverage.

Chapter 2

Getting Started

First, you need to find a workstation on which accounts have been provided for your course. Your instructor should tell you which room(s) to go to, or check the public notices on the first floor of the Computer Science building. If your course is not assigned to the Unix workstations, you should log in on a Windows XP workstation elsewhere in the building.

Find the room where you are supposed to do your classwork, then find a computer that is not in use. The login serves two roles: the banner identifies the hostname of the computer you are using; the login prompt provides authenticated access to the computer. Your login name is the unique name by which you are publicly known to the system. This name is usually just your first or last name, or some combination thereof.

2.1 Logging In

If this is your first computer science course, or if you have gone a semester without taking a computer science course, you will need to complete a quick self-registration. The instructions for this are posted in the computer labs themselves, and are also done with supervision during the CSL Unix Orientation sessions.

After you have completed the registration process for new users, you will be able to login using your own login name and password. Note that when you type your password, no characters will be printed on the screen. This is a security precaution to prevent people from finding out your password as you type it in. If you have an existing account from last semester, you may immediately log in with your previous login name and password. If you do not remember your password you should contact the CSL by sending email

to *lab@cs.wisc.edu* to reset your password.

2.2 The Importance of Passwords

It is very important that you choose a password that is hard to guess and that you keep this password a secret. Your password is the key to your account, and anyone who knows it has complete access to all of your files.

Under absolutely no circumstances should it ever become necessary for you to tell anyone your password. *No one*, including the system administrators, will ever have a legitimate reason to ask for your password. Therefore, if someone does ask for your password, you should never give it to them. Further, you should report anyone trying to get this information to the CSL by sending email to *lab@cs.wisc.edu*. Nor is it ever necessary for you to know someone else's password. In fact, use of someone else's login and password may result in loss of computing privileges for both of you. If you are working on a project with a partner, then you can use to set group access rights to share files. For more information see section 5.6.

If you ever suspect that someone else has used or is using your account, you should notify the CSL (Computer Systems Lab) immediately. To notify the CSL, mail *lab@cs.wisc.edu*. This is very important because the intruder could use up your printer quota, send email as you, or steal your programs and turn them in with his or her name on them. If this happens and this is noticed, the instructor is likely to hold both of the involved parties accountable for the plagiarism, as it may not be clear who stole what from whom.

2.3 Tips on Choosing a Password

The computer system enforces some rules designed to assure high-quality passwords. These rules may make it a bit hard to choose your first password.

The current rules for passwords are:

- Must be at least 8 characters long
- Must contain at least 1 character from each of at least 3 different character classes. The character classes are:
  - lowercase letters
  - uppercase letters

  - numbers
  - punctuation

- Must not appear to be systematic ("abcdef" will be rejected).
- Must not be based on anything that the system knows about you. (name, login, userid, etc.)
- Must not be based on a dictionary word or a reversed dictionary word. A complete word as a substring will cause a password to be rejected.

These rules are based on *crack*, a widely available (and widely used) password guessing program. In addition to the above rules, consider the following:

- Mix UPPER and lower case letters
- Include punctuation marks. Using an exclamation point (!) or a period(.) at the end of your password is less secure, simply because everyone does at the end of phrases.
- One common scheme is to use letters from a phrase. For example, the phrase *choose a good password* might become the password `CagP` (mixing upper and lower case). Of course, this password is not good, because it is too short, has only letters, is easy to guess, and is written in this document. If you do use this scheme, don't use a phrase that is easy to guess (such as your favorite saying or the first line of your favorite song).
- Another common scheme is to start with two or more unrelated words, and abbreviate or mangle them in some manner, so that no part will be in the dictionary. Make sure the two words aren't easily guessable.
- Use symbols to represent some of the words in a chosen phrase.

Bad ideas for your password include the following list, and anything based on ideas from the following list:

- Your phone number or any phone number associated with you, such as your significant other's.
- Birthdays of you or anyone associated with you.
- Addresses of you or anyone associated with you.

- Nicknames of you or anyone associated with you.
- Hometowns of you or anyone associated with you.
- Names of loved ones, pets, etc.
- Any number associated with you (social security, bank account, driver's license, license plate, etc.).
- Any famous personality (rock stars, sports players, teams, or mascots) that you can call you 'favorite' or that people know you like.

2.4 Logging In

After you have successfully entered your login name and password, the system will log you in. By default, you will see three icons on the left side of the screen, and bars or "panels" at the top and bottom of the screen. More detail will be given about these in the next sections.

2.5 X Window System Environment / GNOME

Unix uses the X Windows system to provide a **graphical user interface** (or **GUI**). The default CSL-supported desktop is **GNOME**. GNOME is a powerful, user-friendly, fully-featured desktop environment that is very similar to the interfaces used by Windows or Mac OS. Users who are familiar with these operating systems and GUIs find GNOME very familiar and intuitive.

Once you are logged in, you should see a window in the upper left-hand corner of the screen with something like `[bbadger@demo01] (1)$` inside of it. This is called your **prompt** and means that the computer is waiting for you to type a command. The prompt is in a window called an terminal emulator, specifically **GNOME Terminal**, which emulates a text-only terminal in a windowing environment. More information will be given about this later. Unlike other windowing environments you may be familiar with, Unix is more command-line oriented. This means that the Terminal is one of the most common ways of communicating with the computer. You'll more than likely have a different **hostname** than **demo01** and a different **login** than **bbadger**. This is fine, as this material applies to all CSL Unix machines and all who use them. All the things you see on your screen, as well as all the things you can do to manipulate the image of your screen is

called your environment. This environment has been carefully designed to be the easiest to use during your introductory computing classes.

2.5.1 When You First Log In

After you log in, you should see three icons by default: Computer, [your login]'s home, and Trash. Double-clicking on these icons with the left mouse button will open a GUI file browser to the appropriate place in the file system. The computer will allow you to browse parts of the file system to which you have access, [your login]'s home will let you browse files in your home directory, and Trash will allow you to see files deleted by the file browser and recover them for a short time.

At the top and bottom of your screen are **panels**, which can hold menus, program launchers, and a wealth of other features. By default, the layout of the top panel is as follows. In the top-left corner lie the Applications menu, which allows you to run a variety of programs and change some settings, and the Actions menu, which allows you to search for documents and log out, among other things. Please keep in mind that the CSL does not fully support all of the software available to users.

To the right of these menus are program launchers, which will launch the standard CSL supported web browser, email client, and office software when clicked. The upper-right corner contains a clock and volume control icon.

The bottom panel maintains a list of currently running program windows. Clicking on any given entry in the list will bring the respective window and program to the front of the desktop and make it the focus of the window manager. There is also a button in the lower-left corner that will hide or restore all windows when clicked.

In the lower-right corner of the bottom panel is the **Workspace Switcher**. This powerful tool allows you to have windows on multiple virtual desktops and switch between said desktops. For instance, you could have a web browser on desktop 1, an email client on desktop 2, your programming projects on desktop 3, and other random programs on desktop 4. Simply click on the small desktops in the switcher to change between them. You can even change a window's desktop by dragging it between the screens with the left mouse button.

2.5.2 Window Operations

Manipulating a window's size, shape, and location in GNOME is remarkably similar to Windows and Mac OS. Dragging on any part of the **frame** or **border** with the left mouse button of the window allows you change its size and shape. You can move windows around by dragging on the **title bar** (the bar across the top of the window).

In addition, the upper-right corner of the window contains buttons to minimize, maximize, and close windows. These operations can also be selected from the **window menu** in the upper-left corner of the window.

2.5.3 Copying Text

Another feature of your environment is the ability to **copy text** between windows. This works differently than Windows or Mac OS. By highlighting an area of text, said text gets stored into a memory buffer. You can copy this buffer to another place in the current window or a different window by pressing the middle mouse button (the scroll wheel in most cases).

2.5.4 Root Menu

By clicking on the desktop (anywhere a window or icon does not currently exist) with the right mouse button, a menu appears that allows you to open shell terminals to enter other commands, create directories, program launchers, and files on your desktop, or even change your desktop background. This is called the **root menu**. Any directoies, launchers, and files on your desktop appear in the "Desktop" directory in your home directory.

2.5.5 Panels

By default, you have seen many useful things that panels can have. You can add many useful and fun features to panels by clicking on an empty part of the panel with the right mouse button and selecting "Add to Panel..."

Additionally, you can edit, move, or remove items in a panel by right-clicking on the item in question and selecting the appropriate option. There are other options to create, remove, or modify entire panels. Please be careful when doing this so you do not accidentally delete important customizations or clutter up your desktop.

NOTE: The more customizations you add to your desktop, and the more files, directories, and launchers you add to your desktop, the slower GNOME will become, especially for logging in. It will also count against your disk

quota. While you are free to customize your desktop, please exercise moderation.

2.5.6 Running Programs

Before all the features of the X Window System will seem useful, you will probably want to run an actual program. The easiest way to run a program is to simply click on a launcher in a panel or on the desktop. However, launchers may not exist for all programs you would like to use, and it may not be practical to create all those launchers. The other way, which is easy, is to type the name of the program you would like to run in a terminal (bring up the root menu and select "Open Terminal" to bring up a terminal) and hit [**Enter**].

For example, typing `firefox` at the prompt will run the Firefox web browser. You can use Firefox to access information scattered about the Internet. In particular, the Computer Sciences department has a web site containing much information at `http://www.cs.wisc.edu`. The CSL also maintains a web site at `http://www.cs.wisc.edu/csl`.

2.5.7 Account Status

To check your disk usage, printing quota, and anything in the "Message of the Day" file (also called the **motd**, located in */etc/motd*), open a terminal and run the *Xstats* programs. A typical *Xstats* window can be seen in figure 2.1

2.5.8 Logging Out

Under Unix, it is very important to log out before you leave a computer. If you do not, anyone can come along and access your files, send mail as you, or use your paper or disk quotas.

It is better to exit from all programs before you log out. Some programs, such as *Firefox* or *Thunderbird*, protest the type of shut down caused by you logging out.

When on a workstation with an X environment, to logout, select "Log Out" under the "Actions" menu. Click on OK to verify that you indeed want to log out. Please note that selecting "Restart" or "Shut Down" will merely give you an error and then log you out. Please only use "Log out."

For your own protection, it is important that you remember to logout. Typing `exit` or `logout` in all the windows *will not* log you out of your account. You know you have logged out of your account when the login screen

```
Disk Quota Information:
-----------------------
Volume Name       Quota      Used    % Used      Partition
u.bbadger         200000     125072  62%         65%

Paper Quota Information:
------------------------
User            Quota       Printed
badger          300         24

Message of the Day:
-------------------
==========================================================================
    REMINDER: NO FOOD or DRINK IN THE CS INSTRUCTIONAL COMPUTER LABS
              NEVER POWER DOWN WORKSTATIONS IN THE COMPUTER LABS
==========================================================================
```

Figure 2.1: A typical *Message of the Day* window

reappears. If you don't logout, someone can sit down at the workstation you have abandoned and access all of your files. Be aware that if the system should crash while you are logged in, you will no longer be logged in when it comes back up. Consequently, there is no need for you to wait around for the workstation to restart so that you can logout.

2.6 Machine Availability

All instructional Unix workstations run by the CSL are available in the labs, from approximately 7:00am to 1:00am, seven days a week. The machines are actually on 24 hours a day, so you can connect remotely at any time, but beware that they are rebooted some nights around 3:00am. The CSL's primary objective is to keep these workstations in a usable state all the time. To further this goal, the CSL occasionally needs exclusive access to a machine, rendering it unavailable. Machine downtime will be advertised in the message of the day (*/etc/motd*), which appears when you run *Xstats*, and also in *msgs* with at least 24 hours advance notice.

2.7 Computer Room Etiquette

The computer rooms are a common resource and must be shared by many people, and some of them are under an extreme amount of pressure. To ensure a reasonable working environment and that everyone's equipment stays in reasonable working order, the CSL asks all users of the instructional labs to observe a few simple rules of common courtesy. These include prohibitions against drinking, smoking, eating, and loud music in the instructional labs. If you bring a portable music player, you must use headphones to listen to it so as not to disturb others. These are further detailed in the Account Policy Form which all students will be asked to read when setting up their account. You can review the CSL's policies regarding use of the instructional machines on the web at `http://www.cs.wisc.edu/csl`. *Violation of these rules can result in the suspension of your account.*

2.8 Home Computers and SSH Use

You can access the CSL Instructional Unix workstations from any computer that is connected to the Internet. The best way to do that is with an SSH client. SSH is a secure remote-access program. An SSH client is included

in the WiscWorld software package distributed on campus by DoIT. SSH clients are also available from various sources on the net.

If you have problem getting your home computer set up, the CSL will *not* be able to help you. For this type of problem, go to the DoIT helpdesk. The phone number for the DoIT helpdesk is *264-HELP* or *264-4357*. However, the CSL does maintain an FAQ to help you with home configurations at *http://www.cs.wisc.edu/csl*, but this is as far as the CSL can support home configurations.

The current instructional workstation names, locations, and operating systems are listed on *http://www.cs.wisc.edu/csl*. You can log on to any of the workstations with Unix or Linux listed as the operating system. Further, you can use `best-<computer>.cs.wisc.edu` to select the computer with a name like `<computer>##.cs.wisc.edu` with the lowest load. For example, you could log in to `best-demo.cs.wisc.edu`. If `demo01.cs.wisc.edu` has the lowest load among the `demo##.cs.wisc.edu` computers, you will automatically be logged in to `demo01.cs.wisc.edu`. Finally, logging into `best-linux.cs.wisc.edu` will log you into the instructional Unix/Linux computer with the lowest load.

Chapter 3

Essential Topics

This chapter describes nearly everything you will need to know to successfully complete your course work. It is a good idea to become comfortable with all of the topics in this chapter as early as possible so that you are not struggling with your environment while working on your assignments. The best way to become comfortable with Unix, as with most any computer topic, is to sit down at the computer and experiment. Use the information here as a guide and don't be afraid to try new things. Of course, you must exercise caution with some commands such as *rm*, which removes files.

Each of these sections is a basic introduction to the variety of options available to the described tools. It is almost a trademark of Unix to provide a plethora of options to each available command. To learn more about each tool, consult chapter 4 Getting Help for ideas of where to look. Among other things, this chapter explains the commonly used and verbose *man* program. Some topics are also further covered in Chapter 5, Advanced Topics.

3.1 The Shell

In Unix the **shell** is a program which prompts the user and directs the operating system to do what the user wants. When you login to your account or open a new terminal, you see a prompt like `[bbadger@demo01] (1)$` . This means that the shell is waiting for you to type a command. The shell is not Unix (and neither is X windows). It is merely a user interface to Unix and a program like any other. However, it is essential to know this user interface to become proficient in Unix.

Your user account is configured with *tcsh*, an enhanced version of the Berkeley UNIX C shell, *csh*. If you would like to use one of the many

alternate shells, you can use the *chsh* command to change the shell your account uses.

The shell is a very powerful tool that can be customized. Using just *tcsh*, new commands can be created, old ones altered, input and output files dynamically redefined, and even entire programs written without ever using a compiler. Chapter 5 contains a number of useful hints to get more out of the shell. Consult the manual page for *tcsh* (see chapter 4, Getting Help) for more information on the many features.

3.2 Control Characters

As you've likely encountered, computer keyboards rarely have similar layouts beyond the standard "typewriter" set. Control characters provide many functions such as editing, program control, and output control. Occasionally, a keyboard might not include common special keys. If this is the case, you will need to know a few of the following control characters. These control characters are summarized in table 3.1

To use a control character, simultaneously hold the **CTRL** key and press another key. The combination is usually written with the **CTRL** key being represented as a caret (^). You might notice on some keyboards that backspace produces a `^?` instead of deleting the character to the left. If this happens, you will need to use ^**H** to back up a letter. If you'd like to erase the previous word, use ^**W**, and to erase the whole line, use ^**U**.

3.3 Special Control Characters

There are a number of other control characters that you can type that perform special functions. They are summarized in table 3.1. For example, if your program is caught in an infinite loop, you can kill the running program by typing a ^**C**. You can also kill the program using a ^\. This makes the system create a file for you, called *core*, which is an image of the running program. You may want look at this to do postmortem debugging with the *gdb* debugger in order to find out what went wrong with the program, but you will want to remove the *core* file when you are finished because it takes up a lot of disk space.

If a program is spewing output at you and you want it to pause for a moment, but not kill the program, you can use ^**S** to stop output scrolling and ^**Q** to continue it when you're ready to read further. If you're inputting

Control Character	Function
CTRL-h	erase the last letter on this line
CTRL-w	erase the last word on this line
CTRL-u	erase the current line
CTRL-c	kill a running program
CTRL-\	kill and dump core
CTRL-s	suspend output
CTRL-q	continue output
CTRL-d	End of File (often EOF)
CTRL-z	suspend program

Table 3.1: Common control characters

text to a program, the standard way of indicating the end of the text is with the End of Transmission character **^D**.

3.4 Running Programs

To run a program you need to type in its name. Many programs accept further information on the command line which modify their behavior. Most **commands** take **arguments** and sometimes commands have several **options** or **parameters** available. Most often they are in the arrangement of:

```
command [-parameters] argument1 argument2 ...
```

The brackets around the parameters indicate they are optional.

For example, if you want to see the names of the files in the current directory, the *ls* command is used.

```
[bbadger@demo01] (1)$ ls
```

But if you want to view the contents of your `public` directory while in your home directory, the directory name, `public`, must be supplied as an argument.

```
[bbadger@demo01] (1)$ ls public
```

Many programs also accept flag parameters, indicated with a preceding minus sign, which change the way the program works. These options generally precede other arguments, but this is not always the case. It depends on the the specific program you are using. For example, *ls* doesn't display all the information known about files in a directory. However, if the `-l` option is given, then more information is displayed.

```
[bbadger@demo01] (1)$ ls -l
```

Like most commands, *ls* takes many different parameters, which can be combined together to yield various effects. However, keep in mind that different programs may interpret different arguments as different options. To some programs, `-s` might mean one thing, but to others it probably means something completely different. You should always consult the manual page on the specific program you wish to run for exactly what the various flags do.

3.5 The File System

The file system provides a means to organize and store **files** on permanent storage media (e.g. disks). A **file** is some collection of information. Files may contain, for example, your latest program code, an email, a picture, or a program. The first two of these are usually termed **text** files, while the latter are called **binary** files. Programs are also referred to as **executable** files. To Unix your file is merely a collection of characters. Be careful, though, because binary files tend to lock up terminals and printers if you try to display or print them.

3.5.1 What can I do with a file?

Many different operations are possible on files. In the course of writing a Java program, you will likely want to

`[bbadger@demo01] (1)$ emacs Prog1.java`	edit the program text
`[bbadger@demo01] (1)$ javac Prog1.java`	compile the program
`[bbadger@demo01] (1)$ java Prog1`	and execute it
`[bbadger@demo01] (1)$ lpr Prog1.java`	print the program
`[bbadger@demo01] (1)$ rm output.data`	remove the output file

3.5.2 What's in a filename?

Files must be named so that you can identify them to the system (these names are appropriately called **filenames**). Filenames can be up to about 255 characters, although shorter names are recommended, and can contain many different characters, although we suggest you stick to the standard ones: letters, digits, underscores, dashes, and periods. Some typical filenames are *main.c*, *prog1.C*, *file.o*, *a.out*, and *main*.

Suffix	Meaning
.c	C source code
.cc or .C	C++ source code
.java	Java source code
.html	HTML markup
.o	object code (*not* human readable)
.Z	compressed file (use *uncompress*)
.gz	gzipped file (use *gunzip*)
.tar	tarred group of files (use *tar*)

Table 3.2: Common Suffixes

Note that filenames are case-sensitive; *prog1.c* is different than *pRoG1.C*. This is an important thing to get used to, and may cause some confusion.

Filenames which contain shell metacharacters (that is, characters that the shell interprets as something other than plain text) that also have special meaning to the shell should be avoided, as well as spaces to prevent confusion. For more information see section 3.6.

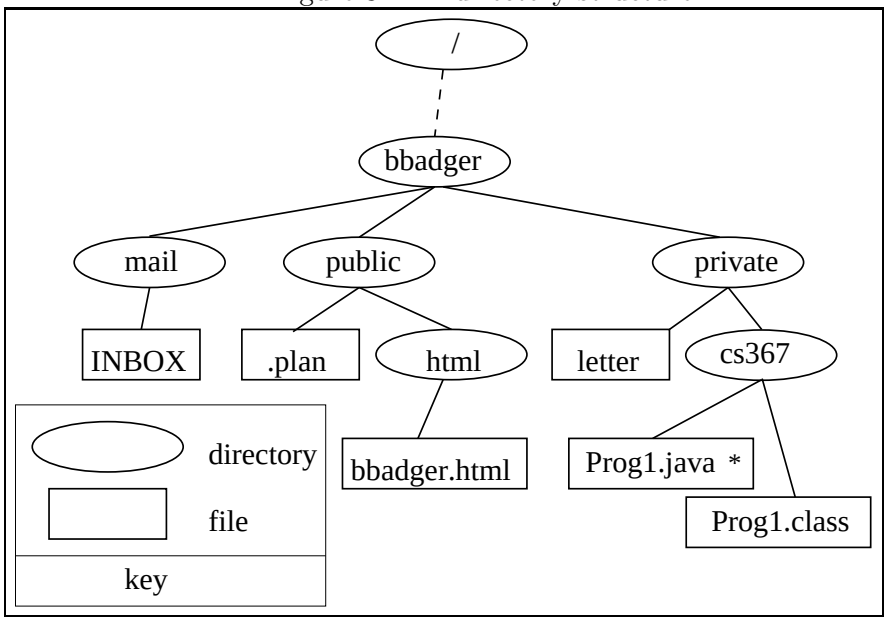
There are several conventions for naming files. For example, you may have noticed above that the file containing the Java program was named *Prog1.java*. The Java compiler expects its input from files whose names end in *.java*. In general, a suffix designates a source file (program text) and the absence thereof designates an executable. Table 3.2 has some of the common suffixes. Note that, unlike in some other operating systems, Unix does not associate a purpose or function based on a filename. Many programs, however, do expect certain suffixes.

The file system separates users' files using a concept called **directories**. Directories are special files that can contain both files and more directories. Each user on the system has a **home directory** in which they may create files and directories. Directories are hierarchically organized; that is, a directory has a parent directory "above" it and may also have child directories 'below' it. Similarly, each child directory can contain other files and also other child directories. Because they are hierarchically organized, directories provide a logical way to organize files. As you read through the next section, refer to figure 3.1.

3.5.3 Pathnames

Files are uniquely named in the file system by specifying the **path** of directories to look in to find the file. For example, the file with the * in figure 3.1

Figure 3.1: A directory structure



is named

```
/u/b/b/bbadger/private/cs367/Prog1.java
```

This is its absolute path name. (The dashed line in figure 3.1 represents skipped levels) Notice that we use the forward slash to separate directories in the pathname.

So that you don't need to type the whole path every time you want to access a file, Unix has the concept of a **current directory**. When you log in, your current directory is your home directory. Files in your current directory may be specified without any pathname attached to the filename. Files in directories below the current directory can be specified with that part of the pathname that begins at the current directory (e.g. if Bob were in his private directory, he could access *Prog1.java* with `cs367/Prog1.java`). This is known as using a **relative path name**.

3.5.4 Navigating the File System

Two commands are useful to navigate the file system tree. They are **pwd** (print (the name of) working directory) and **cd** (change directory). See the examples below.

```
[bbadger@demo01] (1)$ cd                    change directory to your home directory
[bbadger@demo01] (1)$ pwd                   print the name of the current directory
/afs/cs.wisc.edu/u/b/b/bbadger
[bbadger@demo01] (1)$ cd cs367              oops, cs367 is not a subdirectory of
cs367:  No such file or directory           /afs/cs.wisc.edu/u/b/b/bbadger
[bbadger@demo01] (1)$ cd private/cs367      this is what we want
[bbadger@demo01] (1)$ ls                    list the current directory's files & directories
Prog1.java Prog1.class
[bbadger@demo01] (1)$ cd ..                 '..' means "the directory above the current
                                            one" (the parent directory)
[bbadger@demo01] (1)$ pwd                   Now we're back in the private directory,
                                            /afs/cs.wisc.edu/u/b/b/bbadger/private
```

3.5.5 Other's Files

So that you don't have to type the complete path of someone's home directory, the shell provides a shorthand for getting to people's home directories: the ~ (tilde) character. Precede a user's login name with the ~ character and the shell will expand it to the complete path name. In the special case where you do not give a login name after the ~, the shell will expand it to your home directory, making *cd ~* equivalent to *cd*.

```
[bbadger@demo01] (1)$ cd ~cs367-1           cd to cs367-1's directory.
[bbadger@demo01] (1)$ ls
public/ private/
[bbadger@demo01] (1)$ cd private            try to look at a protected directory
private:  Permission denied
[bbadger@demo01] (1)$ pwd
/afs/cs.wisc.edu/p/course/cs367-1           still in this directory
[bbadger@demo01] (1)$ cd ~/private          cd to your own private directory
[bbadger@demo01] (1)$ pwd
/afs/cs.wisc.edu/u/b/b/bbadger/private
[bbadger@demo01] (1)$ cd ~bucky             cd to bucky's home directory
[bbadger@demo01] (1)$ pwd
/afs/cs.wisc.edu/u/b/u/bucky
```

Please note that even though you can list the contents of someone's home directory, you cannot always access all these files. Files are protected using **AFS** and Unix permissions. See chapter 5, Advanced Topics, for more information.

3.5.6 File Management

The following is a summary of common UNIX commands used for file management. Most of the commands are self-explanatory, yet have more advanced options. See See chapter 4, Getting Help, and chapter 5, Advanced Topics, for more information, or consult the man pages.

`cat file1 file2 ...` sends the contents of one or more text files to standard output (usually the screen). Be sure not to *cat* binary files. When binary files are sent to the screen, the terminal tends to lock up.

`less file1 file2 ...` displays the contents of a file a screenful at a time. **SPACE** or **PAGEDOWN** show the next screenful, **b** or **PAGEUP** show the previous screenful, **ENTER** or ↓ show the next line, ↑ shows the previous line, **?** brings up a help list, and **q** quits viewing. The name *less* is a pun on *more*, which is another pager with fewer features. Thus, *less* is *more* (but better, more than *more* if you will).

`cd path` changes the current directory to *path*.

`ls [dir]` lists the files in *dir* or the current directory if no argument is given.

`pwd` prints the name of the working (current) directory.

`cp source destination` copies a file from *source* to *destination*

`mv source destination` moves a file from *source* to *destination*.

`rm file` removes a file.

`mkdir dir` creates a directory of the name *dir*.

`rmdir dir` removes a directory of the name *dir* if it is empty.

3.5.7 Disk Quotas

There is a limited amount of disk space available, and this space must be shared by all of the users. In an attempt to ensure that all students have adequate space to complete their assignments, a disk quota system has been put into effect. The quotas allotted to students should be sufficient to carry out their programming assignments. However, if you create many extra files or do not remove unnecessary files, you may exceed your quota.

When you login to your account, the quota command, *fs listquota*, is automatically run for you and displayed in the *Xstats* box. This command returns some statistics on your current disk usage:

```
[bbadger@demo01] (1)$ fs listquota

Volume Name                    Quota      Used %Used   Partition
u.bbadger                     200000    125072   62%         65%
```

Each user is assigned a **volume** on the disk to store their files. Several volumes are stored on each disk **partition**. The **Quota** field shows your disk quota, that is, the amount of disk space, in kilobytes, that you are alloted. The **Used** field indicates how many kilobytes of disk space you are currently using. The **%Used** field shows what percentof your quota you are currently using, and the **Partition** field shows what percent of the entire partition is currently being used by all of the volumes stored on it. Do not worry if you get a warning message saying the partition is nearly full. You will still have the same alloted quota.

It is important to make sure that you do not exceed your disk quota. If you should go over your disk quota (i.e. the Used field equals or exceeds the Quota field), the quota system will not allow you to create any new files until you reduce your disk usage. You can run the command *fs listquota* at any time to see your current disk usage.

Guidelines for controlling your disk usage

- Remove *core* files. When your program exits abnormally, many times a memory image of your program is saved into a file called *core*. When this happens, you may see a message such as "Segmentation Fault - core dumped". A core file can be quite large (the size of all the memory alotted to your program). They can be useful for debugging, but remove them as soon as possible.

- Remove *cache* files. Your web browser's cache can take up logs of space. If you use Firefox, Select "Tools", "Clear Private Data", and make sure "Cache" is selected when you click "Clear Private Data Now".

- If you don't know what files are taking up your disk space, the command *du* displays the size of all of your directories and files. Large files can be seen using the CSL-written-and-supported program *lff*. You can type *lff ~* to see a list of large files and directories in your

Metacharacter	Function
;	seperates multiple commands on the same line
?	matches any one character in a file name
*	matches zero or more characters in a file name
&	runs the preceding command in the background
\	nullifies the special meaning of the next metacharacter

Table 3.3: Common Metacharacters

home directory. Consult the manual page (type *man lff*) for more information.

- Remove executable files that you are not currently using. Executables can always be regenerated if you have the source code.
- If you are continually exceeding your quota, consider saving part or all of your directory to CD-ROM. Instructions are available on the web.

3.5.8 Restoring Deleted Files

Every night around midnight, a "snapshot" is taken and saved of all files and directories in your account. If you accidently remove a file, it may be possible to get it back if it existed at the time of the snapshot.

To recover a file, type the command `recover list` to list the files available in the snapshot of the current directory, and `recover fetch filename` to recover the file *filename*.

If the file is not in the snapshot, go to `https://www-auth.cs.wisc.edu`, click on "Continue to Web Forms" and fill out the "Restore Request Form" to request that the CSL restore your file from backups.

3.6 Shell Metacharacters

There are a number of characters that have special meaning to the shell, which is what makes them metacharacters. Note that meta- is a generic prefix assuming different meanings for different programs. Be careful not to confuse their meanings or confuse them with metakeys. Common *tcsh* metacharacters are (! $ ^ & * ( ) ~ [ ] \ — { } ' " ; ¡ ¿ ? [space] [tab]). The metacharacters are explained further in the man page for *tcsh*. Table 3.3 contains very commonly used metacharacters.

As an example,

```
[bbadger@demo01] (1)$ ls file.*
```
run on a directory containing the files *file*, *file.c*, *file.lst*, and *myfile* would list the files *file.c* and *file.lst*. However,
```
[bbadger@demo01] (1)$ ls file.?
```
run on the same directory would only list *file.c* because the ? only matches one character, no more, no less. This can save you a great deal of typing time. For example, if there is a file called *california_cornish_hens_with_wild_rice* and no other files whose names begin with 'c', you could view the file without typing the whole name by typing this:
```
[bbadger@demo01] (1)$ more c*
```
because the `c*` matches that long file name.

Filenames containing metacharacters can pose many problems and should never be intentionally created. If you do find that you've created a file with metacharacters, and you would like to remove it, you have three options. You may use wildcards to match metacharacter, use the \ to directly enter the filename, or put the command in double quotes (except in the case of double quotes within the file name, these must be captured with one of the first two methods). For example, deleting a file named '`*|more`' can be accomplished with:
```
[bbadger@demo01] (1)$ rm ??more
```
or
```
[bbadger@demo01] (1)$ rm \*\|more
```
or
```
[bbadger@demo01] (1)$ rm ``*|more''
```

3.7 World Wide Web and Mozilla Firefox

The CSL's currently supported web browser is Mozilla Firefox. It is part of the suite of Mozilla software. This brief introduction will highlight some of Firefox's unique features and how to manage your own Firefox settings.

To start Firefox, type *firefox* at the command prompt. If this is the first time you have run Firefox under you Computer Science account, it may take a while to initialize your settings. Once Firefox appears, you can see the default layout, with menus, browse buttons, the address bar, and some bookmarks at the top, as well as the status bar at the bottom, much like other web browsers. There is also a built-in interface to search engines in the upper-right corner of the Firefox window. Simply enter something in this box and press Enter to query the appropriate search engine.

Firefox supports tabbed browsing. That is, multiple pages can be open in the same window. When multiple pages are open, you will see tabs representing each page just below the bookmarks. Click on a tab to go to that page. To open a new tab in an existing window, select "New Tab" from the "File" menu or click on a link with the right mouse button and select "Open Link in New Tab".

You can manage your Firefox settings by selecting "Preferences" from the "Edit" menu. Here you can change how much space to use for the local cache, how many days to keep history, your home page, and many other options.

Please explore the Computer Science Department website and the CSL website to find more information. Often, answers to common questions are on these pages.

3.8 Electronic Mail and Mozilla Thunderbird

Email is available on all student accounts. Your Computer Science Department email address is your login name followed by @cs.wisc.edu. Please note that this email is different than your DoIT WiscWorld email account. The CSL's currently supported email client is Mozilla Thunderbird, another part of the suite of Mozilla software, and will be briefly outlined in this section.

3.8.1 Reading Mail

To start Thunderbird, type *thunderbird* at the command prompt. If this is the first time you have run Thunderbird under you Computer Science account, check to ensure that the automatic account creation procedure was completed successfully (there should be a <login>@cs.wisc.edu entry in the left pane - if this is not present, consider mailing lab@cs.wisc.edu from a different account to troubleshoot the problem). Click on the plus or minus sign to the left of the account entry to expand or collapse the folder hierarchy. When the view is expanded, click on "Inbox" to bring up a list of messages which should then appear in the upper right pane. To read a message, click on the message in the upper right pane; the text of the message will subsequently appear in the lower right pane. If you desire to reply to a message, highlight the message and click on the "Reply" icon in the toolbar.

3.8.2 Composing Mail

To compose a message in Thunderbird, click on the "Write" icon in the toolbar. This will open a blank e-mail message. Enter the address of your correspondent in the "To:" field. Include a subject at your discretion. Finally, type the text of your message into the message body beneath the subject field. When you are satisfied with your message, click on the "Send" icon. You will be automatically prompted to spellcheck the document. After you have completed the spellcheck, click "Send" one more time to send the message on its way.

3.8.3 Managing Mail

Mail spools can grow quite large if not managed and may count significantly against your disk quota. Should you desire to delete a message, highlight the message and then click on the "Delete" icon in the toolbar. Note that this does not actually remove messages from your mail spool, it merely marks the messages as "to be deleted." To delete messages, either right-click on a folder and select "Compact this Folder" or select "Compact Folders" from the file menu. Please note that this completely removes messages from a mail spool. Unless they were available when backups were made, they are lost forever.

3.9 Editing Files with Emacs

To edit your programs, you will need to use an editor. An editor is a little like a word processor, except that it doesn't know about concepts such as fonts or styles. It only understands plain text. The better editors have features which make it easier to write code. The editor we suggest is called GNU Emacs.

Emacs provides a menu-based interface for editing text. The first-time user will appreciate the menus of common features. If you are familiar with popular word processing programs, Emacs should seem fairly easy to navigate. It is important to remember, though, that Emacs is *not* a word processor. A word processor has functions for writing, printing, and saving text with enhancements such as bold text, special paragraph formatting, and pictures. Emacs is a **text editor**. It is suitable for editing code and other plain-text files.

The largest difficulty is learning that Emacs divides everything into **buffers**. If you access the help pages from the help menu while editing

a text file, you will need to either kill the buffer with the help pages, or select the text file buffer (under the Buffers menu) to edit the text file again. The best way to learn Emacs is to experiment, read the help, and use the tutorial.

- Special functions are accessed with either the **control** key or **meta** key. Control functions are indicated in the form of **C-x**, which means hold down the control key while pressing **x**. Meta functions are indicated in the form of **M-x**, which means hold down the meta key (the 'alt' key on most PC keyboards) while pressing **x**. If there is no meta key on your keyboard, it can be simulated by pressing and releasing the **ESC** key, and pressing the other key. **M-x command** means press **M-x** and then enter the *command* followed by return. If you cannot remember the entire command name, pressing TAB will complete the name, and '?' provides a list of possibilities.

- If you make a mistake entering a command, **C-g** will cancel it. This only works while entering it, not afterwords.

- If you make an editing mistake, you can select **Undo** from the **Edit** menu or type the equivalent **C-x u**.

- If you invoke Emacs from the shell on an terminal emulator, the shell will be in use until you quit Emacs, unless you tell it to run Emacs in the background. This is done by appending an ampersand on the command line, i.e. `emacs &`.

- If you start Emacs in a text-only environment (From a remote shell, for example) and wish to exit, use the sequence **C-x C-c**.

- Emacs allows you to create multiple 'windows' within its alloted window to view any of its buffers. **C-x 2** creates two windows, likewise, **C-x 1** returns you to one window.

- Emacs also allows you to use multiple frames running the same program. To open a file in another frame, choose *Open File in Other Frame* from the File menu, or type **C-x 5 C-f**.

3.10 Printing

All instructional accounts have access to the printers in room 1359. The print spool that services those printers is called **laser**. To send a file to the

printer, you can use either *lpr* or *print*. *print* **only** works with text files and includes a banner across each page indicating the filename, page number, time, and username. If you send a postscript or graphics file with the *print* command, you will use up your paper quota by printing useless gibberish. Use the *lpr* command instead.

```
[bbadger@demo01] (1)$ lpr file
[bbadger@demo01] (1)$ print file
```

Your print job will be sent to the first available printer. All files sent to the printer are put in a queue and printed in order of arrival. You can use the command *lpq* to display the queues of all the laser printers and determine to which your job was sent. Also there are 2 print monitors in 1359 which display the status of the two printers.

You also have the ability to remove a job while it is still in the queue with the *lprm* command. The following will remove *all* of your print jobs.

```
[bbadger@demo01] (1)$ lprm login
```

Where *login* is your login name.

Students are given a 150 page paper quota per class. This amount of paper should be plenty to cover all printing needs for one class, but if there is a need for more paper, it can be purchased at the Computer Systems Lab (room 2350) for a small fee (payable by check or money order only). Plan ahead if you start to run low on paper, as the CSL is only open during normal business hours, Monday through Friday.

To see how much paper you have and how much you have used, use the *lpquota* command.

Note that it is your responsibility to check that the printer is working before you send your files. That is, if you send your printout to a printer which is low in toner or printing streaks, your paper quota will not be reimbursed. Please send mail to `lab@cs.wisc.edu` if you notice a printer with problems.

Chapter 4

Getting Help

4.1 Unix Manual Pages (the *man* command)

By far the most comprehensive source of unix command information is the online manual. In fact, the entire Unix Programmer's Manual is stored online. The program to access the manuals is called *man*. The argument passed to it (whatever else you type on the command line after the command name) is the topic you want information on. For example, if you wanted *man* to tell you about itself, you would type:

`[bbadger@demo01] (1)$` *`man man`*

This will show the manual page on the *man* program itself. Be aware that man will only explain something to you if you can ask for it by name. Unfortunately, there are many occasions when you don't know what Unix calls the program you need to run. You may use *apropos* or *man -k* to find out what the system knows about some subject. For example,

`[bbadger@demo01] (1)$` *`apropos compiler`* (or *`man -k compiler`*)

```
gcc(1) in std-14 - GNU project C and C++ compiler
g77(1) in std-14 - GNU project Fortran 77 compiler
gcj(1) in std-14 - Ahead-of-time compiler for the Java language
javac(1) in std-14 - Java compiler
jikes(1) in std-14 - java source to bytecode compiler
perlcompile(1) in std-14 - Introduction to the Perl Compiler-Translator
xsubpp(1) in std-14 - compiler to convert Perl XS code into C code
compile_et(1) in std-14 - error table compiler
ccmakedep(1) in std-14 - create dependencies in makefiles using a C compiler
gfortran(1) in sys - GNU Fortran 95 compiler
rpcgen(1) in sys - an RPC protocol compiler
```

Section	Contents
1	Programs (commands)
2	System Calls
3	Subroutine libraries
4	Hardware
5	Configuration Files
6	Games
7	Miscellaneous
8	System Administration

Table 4.1: Sections of Man Pages

```
yacc(1) in sys - yet another compiler compiler (DEVELOPMENT)
uil(1) in sys - The user interface language compiler
checkpolicy(8) in sys - SELinux policy compiler
zic(8) in sys - time zone compiler
...
```

apropos and *man -k* show you all the various subjects that the online manual knows concerning the topic compilers. The words *gcc, javac,* etc. are items that *man* knows about. The parenthesized number that follows the command name indicates the section of the manual where the topic can be found. You need to know this because often the same topic appears in more than one section of the manual, and you need to be able to specify which section you are interested in. The various sections and their uses are summarized in table 4.1. Be aware that these commands must search through a lot of data and may be slow.

Sections can also be divided into subsections. These subsections are denoted by one letter, and indicate which library the routine can be found in (c for compatibility, f for Fortran, m for math, etc.) You may notice that topics may appear in more than one place. To distinguish which page you want to see, you precede the name with the section. Some examples are:

```
[bbadger@demo01] (1)$ man 8 zic
[bbadger@demo01] (1)$ man 8v tic
```

Note that there is an intro page for each section (1, 2, 3, 3f, 3m, etc.) Thus if you would like to know more about math subroutines, type

```
[bbadger@demo01] (1)$ man 3m intro
```

Commands for viewing the man pages are identical to those for *less*. Press **h** for a summary of *less* commands.

4.2 Other Sources of Help

A plethora of information (including this document) is available on the web, and includes, among other things, information about the Computer Sciences Department.

You will often find that asking the person sitting next to you is the quickest way to an answer. However, the students around you need to work on their projects, too. Depending on what time you are in the lab, you may find **consultants**, who circulate around the terminal rooms (look for the "consultant on duty" signs they put up, usually on top of the monitor they're using or on the desk near them). They are teaching assistants who hold office hours and can answer UNIX related questions. Also, the CSL staff offers **Unix orientations** at the start of each semester.

There are several books on Unix are available at the local bookstores. The world wide web (especially Google) is also a great place to find answers, tutorials, references, and forums regarding Unix. The CSL maintains a web page, complete with documentation, at *http://www.cs.wisc.edu/CSDocs*.

Course-related questions are best handled by your TA or professor. This includes things like the locations of data files or specific libraries you need to link into your programs. The CSL staff are not programming consultants and should not be used as such.

There are some matters, however, that only the CSL staff can solve. These primarily pertain to administration and security, such as a nonexistent account or if you think that someone else knows your password. If you think there is a problem with the system software (as opposed to your own program), or a problem with the hardware (such as a hung workstation or broken keyboard), send mail to *lab@cs.wisc.edu* with an explanation of the problem. Be sure to fully describe the problem as best you can or your problem may not be understood. Mail to lab will be forwarded to the correct party to deal with your difficulty.

Chapter 5

Advanced Topics

This chapter summarizes several of the more powerful UNIX tools and features of the instructional machine environment here at the University of Wisconsin Computer Science Department. Many of the following tools are simple yet powerful time savers. This is still only an introduction to the enormous quantity of available features. Do not hesitate to play around and further explore the UNIX environment.

5.1 TAB Completion

In *tcsh*, *bash*, and many other shells, filenames and commands can be completed or partially completed automatically by pressing the **TAB** key. This is commonly known as **TAB completion**. Pressing TAB after typing in a shell prompt will complete the filename, directory name, or command name as far as it can (or completely if it can be uniquely determined from the currently typed letters).

For instance, if you are in your home directory, type `cd pr` then press the TAB key. Since no other files or directories in your home directory (by default) begin with "pr", the shell can determine that you mean to type "private" to refer to your private directory. You can then continue entering arguments into the command, or you can press Enter to execute the command.

In addition, TAB completion can be used to partially complete names. Suppose that there are 3 subdirectories in one of your directories for each of your programming assignments, named program1, program2, and program3, and nothing else. While in this directory, you can type `cd p` and press TAB and the shell will complete the command as far as "program," however it

does not know which program directory you mean. You will have to finish the directory name yourself.

Please note that tab completion can be done with most every command. The *cd* command was merely used as a common example. If you type `em` at a prompt and press tab, "emacs" will be completed since the shell can determine that no other command begins with em.

However, TAB completion has its limitations. In the first example, if you typed `cd p` and pressed TAB, nothing would get added to the command and the computer would likely beep at you. This is because the file you want to enter cannot be uniquely determined by the shell. It does not know whether you want to choose your **p**ublic or **p**rivate directory. You need to give the shell more information before TAB completion will work.

To list all possible completions of a command or file, if you use *tcsh* (the default shell), type **CTRL-D** after partially completing the command. If you use *bash*, press TAB twice instead. If you type `cd p` in your home directory and pressed CTRL-D (or TAB twice), a line will appear with "public/" and "private/" indicating the two possible completions. As always, please consult the manual page of your shell for more information and specifics about how it handles TAB completion.

5.2 I/O Redirection

Many programs in UNIX will by default read their input from the user's keyboard and write their output to the terminal. If you write a C program that uses *printf* and *scanf*, the program will act this way. This is convenient for debugging, but what about when it comes time to actually test it with your TA's data? In most systems, you would have to recompile your program. With UNIX, you can dynamically redirect the normal input and output streams right on the command line without ever touching the program's code. For example:

```
[bbadger@demo01] (1)$ prog1 < test-data
```

runs *prog1*, taking its input from the file *test-data*. The output will still be printed on the user's terminal. On the other hand:

```
[bbadger@demo01] (1)$ prog1 < test-data > results
```

still reads the input from the file *test-data*. Also, the output is sent to the file *results*. For a slight variation, the command:

```
[bbadger@demo01] (1)$ prog1 < test-data >> results
```

would function the same, except the output is *appended* to the file *results*, instead of replacing whatever was there.

5.3 Pipes

Suppose a command produces a large amount of output that you want to search through using *less*. You could produce a file from the command as described above and run less on that file. Or, you could use a **pipe**. Pipes allow you to redirect one program's output directly into another. For example:

```
[bbadger@demo01] (1)$ apropos compiler | less
```

will take the output of *apropos* and pass it to *less*. Then, you can use the *less* program to search through the output. Note that the pipe uses a vertical bar (|), not a slash (/).

Be aware that this does not create a file because the data is piped directly from one program to another. If you would like to create a file from this data, you will have to use any save features of the program to which you pipe.

Further be aware that not all programs respond the same way to pipes. Some programs behave erratically if data is piped directly to them. Consult *man* pages for more info.

5.4 Passwords

The *passwd* command allows you to choose a new password. Refer to section 2.3 for more information about passwords. If you forget your password, you must show up in person to the CSL with your student ID. The CSL will never tell you your password, you must create a new one.

5.5 Dotfiles

Dotfiles are the generic term for the configuration files for various programs. These are usually plain text files with a specific format. The most common form for a dotfile name is .<program name>rc. For example, one of *tcsh*'s configuration files is called *.cshrc*. These files will not show up with the ls command unless you use the `-a` or `-A` options. The *man* pages for programs will have more information about their dotfiles and their format.

5.6 File System

It is possible to allow specific users to access specific directories within your home directory. You have hopefully already noticed the peculiar properties

Right		What it does
r	read	allow user to look at files in the directory
l	lookup	a user with this right may list a directory, look at an ACL or access subdirectories.
i	insert	allows user to add files to a directory.
d	delete	allows files to be removed by user.
w	write	allows files to be written and modified by user.
k	lock	allows advisory file locking.
a	administrator	allows user to change ACL. We do not advise giving other users the administrator right or removing your own administrator right.
shorthand notations for common combinations		
all		rlidwka
write		rlidwk
read		rl
none		*removes entry*

Table 5.1: AFS permissions

of the directories *public* and *private*. Any file or directory you store in *public* is readable by anyone and nothing in *private* is readable by anyone but yourself. This is achieved through the **Access Control Lists** (**ACLs**) for each directory. Each directory's ACL defines permissions for all files in that directory. Subsequent directories created in that directory will initially have the same permissions as their root directory. You can view the ACL of a directory with the command *fs listacl*. For example:

```
[bbadger@demo01] (1)$ fs listacl public

Access list for public is
Normal rights:
  system:administrators rlidwka
  system:anyuser rl
  bbadger rlidwka
```

Each of the characters on the right are abbreviations for the permissions summarized in figure 5.1

To add an entry to an ACL, use the *fs setacl* command:

```
[bbadger@demo01] (1)$ fs setacl <directory> <user> <permissions>
```

For example, if the user "bucky" is your partner on an assignment you are working on in directory called 'project', you'd type

```
[bbadger@demo01] (1)$ fs setacl project bucky write
```

5.7 Communication

finger user@host displays the finger information for *user* at the machine *host*. Most finger servers display the files *.plan* and *.project* if they exist in their home directory. Note that these files must be publicly readable.

who lists users logged into your machine.

5.8 Miscellaneous Commands

grep string file1 file2 ... displays all the lines in the files that contain 'string'

cal month year prints a calendar for the corresponding month and year.

ispell file interactive program that checks spelling in *file*

script filename records all screen I/O into *filename* until you type `exit`. Note that *script* will overwrite the file if it exists.

Chapter 6

Quick Reference

This chapter is a quick reference to commands listed elsewhere in the document.

file commands

`cp` *source destination*	Copy the file from *source* to *destination*.
`mv` *source destination*	Move the file from *source* to *destination*.
`cd` *[path]*	Change the current directory to *path*. If *path* is not specified, change to home directory.
`pwd`	Print the working directory.
`ls`	List the files in the current directory.
`less` *file*	show the contents of *file*

filesystem commands

`fs listquota`	List your current disk usage.
`fs listacl`	List the Access Control List (ACL) of the current directory.
`fs setacl` *path user rights*	Set the Access Control List (ACL) of *path* for *user* to *rights*. See table 5.1 for the list of rights. The most important rights are the aliases *read*, *write* and *none*

help commands

`man` *command*	Get help for *command*
`apropos` *topic*	List command dealing with *topic*

communication commands

`finger` *user@host*	displays information about *user* on the machine *host*
`who`	lists users logged in to your machine

miscellaneous commands

`grep` *string file1 file2 ...*	displays all the lines in *file1 file2 ...* that contain *string*
`cal` *month year*	display a calendar for the specified *month* and *year*
`ispell` *file*	check the spelling of a document
`script` *file*	send all screen I/O to *file* until `exit` is typed.