Athena Workstation

Bill Bryant
Mark Moormans
Sharon Belville
Brandy Bechtel
K. L. Lieben
MIT Project Athena
Revision C
September 6, 1988

Revision C: September 6, 1988. Bill Bryant, Mark Moormans, Sharon Belville. Updated document to reflect changes inherent in the 6.0C release. Added an overview of the Athena system. Reorganized other chapters.

Revision B: February 2, 1988. K. L. Lieben. Updated document to reflect changes to 5.5T release.

Version A: September 11, 1987. Bill Bryant, Jon Freeman, Brandy Bechtel, and K. L. Lieben. Amalgamated information from previous Athena documents, including ***Essential Athena, Essential Unix, Essential X,*** and ***Essential Workstation***.

1. Introduction

This document provides an introduction to Project Athena's workstation environment. Hopefully you will be able to get onto the system and get started as quickly as possible. You will learn how to:

- Login to a workstation.
- Use the Unix operating system.
- Use the X window system.
- Back up files to floppy diskettes.

This document also has three Appendices, which provide additional, but not *essential* information:

- **Appendix A** covers the Unix file protection scheme. Read this appendix if you want to share files with other users.
- **Appendix B** describes how to access course software.
- **Appendix C** describes some of the bells and whistles you can use to customize your workstation's user interface.

This document assumes that you have read the free document **Athena Basics** and the document **Essential Athena** and that you have registered for your Athena account (remember that it takes anywhere from twelve to twenty-four hours for the registration to take effect).

When you feel that you have a handle on the information covered in this document, you will probably want to read **Essential Emacs**, **Printing from an Athena Workstation**, and **More Unix: the C Shell**. These documents may be purchased at Graphic Arts in Building 11, or in the Student Center.

You can also read these documents at a workstation. Use the following commands to access the *athenadoc* locker:

```
athena% attach athenadoc
```

```
athena% cd /mit/athenadoc
```

The *athenadoc* directory contains a **README** file that explains how to use the **viewdoc** program to read the online documentation. Use the **more** command to examine the *README* file:

```
athena% more README
```

When using the *more* command, press the space bar to go to the next page of text.

1.1 Getting Help

The Unix system comes with an extensive, but often obscure online documentation system. You access this system with two commands: **apropos** and **man**. The *man* command displays a command's **Unix Manual page**, so it's only useful if you already know the name of the command you need help with.

If you don't know the name of a command but you have the command's function in mind, use *apropos*. For instance, suppose you wanted to remove a file, but couldn't remember the name of the command. Use *apropos* as follows:

```
athena% apropos remove
atrm (1)                - remove jobs spooled by at
colrm (1)               - remove columns from a file
deroff (1)              - remove nroff, troff, tbl and eqn constructs
flock (2)               - apply or remove an advisory lock on an open file
insque, remque (3)      - insert/remove element from a queue
lprm (1)                - remove jobs from the line printer spooling queue
rm, rmdir (1)           - remove (unlink) files or directories
rmdir (2)               - remove a directory file
rmdir, rm (1)           - remove (unlink) directories or files
strip (1)               - remove symbols and relocation bits
unifdef (1)             - remove ifdef'ed lines
unlink (2)              - remove directory entry
unlink (3F)             - remove a directory entry
unmount (2)             - remove a file system
```

1.1.1 The Athena User Consultants

Athena User Consultants are also available to help answer questions about *Athena-supported* software and hardware. Call the Consulting Hotline (3-4435) for information concerning the consulting schedule.

Consultants can also be reached via the **On-Line Consulting** System (**olc**). To use the *olc* system, type **olc** and follow the instructions.

1.1.2 The info Locker

This locker contains information about many different aspects of Athena, including printer status and locations, cluster status, Athena minicourse schedules, and a list of available software and how to access it.

Use the *info* locker as follows:

```
athena% attach info
info: NFS mounted EUTERPE.MIT.EDU:/mit/lockers/info on /mit/info
athena% cd /mit/info
```

Now use the *more* command to read the *info* locker's **README** file:

```
athena% more README
```

2. An Overview of the Workstation and the Athena System

Project Athena's computing environment is a workstation-oriented one. Just walk into any of the public clusters and you'll see dozens of workstations sitting on the tables.

One of the nicest features of Athena is that you can sit down at any public workstatation and access any Athena program. And you can access your own private workspace from any workstation. You aren't tied to any particular machine or set of machines.

How does the system work? The following paragraphs discuss the Athena environment's subsystems and describes how they depend upon one another to support the workstation system.

2.1 The Workstation

Each workstation has a display monitor, keyboard, mouse, and a system unit. The system unit contains the workstation's Central Processing Unit (CPU), its main memory, and a hard disk drive that provides storage for some of the Athena system's software.

The system unit also contains a **removable media device**. In most instances this device is a floppy disk drive, but there are some workstations equipped with streaming tape devices.

Athena workstations are single-user computers. When you login to one, its computing resources are exclusively yours until you logout.

2.2 The Unix Operating System and the Shell Program

The **Unix Operating System** controls the workstation's hardware. It manages the workstation's CPU and memory, its hard disk drive, and its removable media device. The operating system is the software that gives you control of the workstation.

You issue commands to Unix via a program called the **shell**. Unix starts a shell program for you when you login to the workstation. Unix is ready to take commands when the system displays the shell prompt, which looks like this:

```
athena%
```

You type a command, press the **Return** or **Enter** key, and the shell forwards the command to the operating system. When Unix has taken the command, the shell displays a new prompt to indicate that it is ready to take a new command.

2.3 The X Window System

The **X Window System** is a group of programs that let you divide your workstation's display into one or more rectangular **windows**. Each window operates like an independent computer terminal, so you can work on several things simultaneously. You could write a paper in one window, format it in another window, and look at the formatted version in another.

X makes the workstation a very powerful instrument. After you create the windows you want to use, you can move them around, shuffle them above or beneath one another, and make them any size you want. And you can remove windows from the display at your pleasure.

When you sit down to use a workstation, it creates a window for you to login to. After you have logged in, Unix starts a shell within that window that waits for your commands. You then have the option of creating other windows and running different programs within them.

2.4 The System File Servers

Once you login to a workstation, you have access to many different programs. You can use a text editor to create text files such as papers or programs, a text formatter to format papers and reports, and the electronic mail system to send messages to friends.

Most of these programs aren't actually stored on the workstation because the workstation's hard disk doesn't have enough storage space to hold all of the Athena software. Instead, many programs are stored on large ***file server*** computers. You can access the progams because the workstation is connected to the file servers by a ***network***.

Suppose for instance that you want to use the *Emacs* text editor to write a paper. You type the *emacs* command at the keyboard. The workstation contacts the appropriate file server machine and copies the *emacs* program from that machine into its own computing space. Only then can you use the program to write your paper. Note that this copying operation happens without your having to worry about it. You just type the command, the workstation takes care of the rest.

2.5 Your Personal File System

When you login to a workstation, you gain access to Athena's software, and you gain access to your own personal storage space, the space where you store your work. This space is your personal file system **locker**.

Your locker is a file system that exists "somewhere on the network," on a file server machine. You don't have to worry about the file server's exact location because the workstation finds it and connects to it when you login.

2.6 What Really Happens When You Login

If you sit down at a workstation intending to login, you see one of the following things on the display monitor:

- The screen is dark and displays the message *Hit any key to start.*
- The screen displays a window with the **login prompt**.

In the first instance, the workstation is **deactivated** and needs to be **activated** before you can login. In the second instance, the workstation is activated and is ready for you to login.

Let's assume for the sake of argument that the workstation is deactivated and its screen is displaying the *Hit any key to start* message. Here's what happens when you press "any key" to start:

1. The workstation connects itself to the system file servers. This happens first because some of the programs needed to continue the activation and login processes are on the system file servers.
2. The workstation starts the X window system **server program**, then uses the **xterm** program to create the login window.
3. You enter your username and password.
4. The login program logs you in, and as part of this process, finds your locker and connects it to the workstation.
5. When the login process has finished, Unix starts a shell for you.
6. When the shell displays its **athena%** prompt, the workstation is ready to take commands.

In activating and logging into the workstation, you have put several of Athena's subsystems to work. The workstation can't really serve you without them.

3. Logging In and Logging Out

Logging in is the process of identifying yourself to a workstation and being given the right to use it. You get access to your personal files upon logging in successfully.

When you have finished a workstation session, you must logout. If you forget to do so, the next person who uses the workstation has access to your personal files and can cause all kinds of mischief in your name. **Don't forget to logout!**

3.1 Logging In

Walk up to any idle workstation. If someone has just finished using its, its screen will display a login window and you can login right away. Otherwise the workstation has *deactivated* itself; its screen is dark and displays the message ''Hit any key to start.''. Press the **Return** or **Enter** key to begin.

The words ''Activating workstation...'' appears at the bottom of the screen. More messages appear, then the screen turns into a gray mesh with a small capital **X** in the center. Finally the workstaton creates a login window that displays a message similar to this one:

```
Project Athena (E40-342-19.MIT.EDU:ttyv0, 4:34pm on Fri, 16 Sep 1988)

login:
```

Type in your username in lowercase and press **Return** or **Enter**. When the workstation prompts for your password, enter it in the same way. Notice that the workstation does not display the password as you enter it.

Sometimes it seems like forever for the login process to complete. If all goes well, the *login* window eventually displays a message telling you that the system has started a window manager. The screen may also display a *message of the day* message. The login process is complete when the window displays a shell prompt:

```
athena%
```

You can now enter commands.

3.2 Logging Out

To logout, you must be in the *login* window. Logout by typing **logout** in response to the shell prompt.

A few problems may arise when you attempt to logout. If Unix says ''There are stopped jobs.'' and doesn't log you out, type **logout** again and Unix will give up and let you go. If Unix says ''Not login shell.'', and you are typing commands *in your login window*, type **exit**, and then type **logout** again. Continue this procedure for as long as Unix continues to say ''Not login shell.''.

3.3 *login* Problems

The workstation will not let you login if you mistype your username or password. If you mistype your username, the *login* process may return the following message:

```
Principal unknown (kerberos)
```

If you mistype your password, *login* returns this message:

```
Password incorrect
```

Be sure to type your username and password exactly as you gave them to the registration program; don't insert blank spaces, and remember that there is a difference between lower case and capital letters. To try again, hit **Return** or **Enter** until the *login:* prompt reappears. If you still can't login, ask a consultant for help, or contact the Athena Accounts Administrator at **253-1325**.

Suppose you try to type your username and your typing does not appear in the login window. Make sure that the mouse cursor is in the window. The mouse cursor is the thing that moves when you wiggle the *mouse* attached to your workstation. *The mouse cursor must be inside the login window for your typing to have any effect.* When the mouse cursor is inside the login window, it looks like a capital **I**; otherwise, it looks like a hollow capital **X**.

3.4 Fixing Typos, Bailing Out

When you see the Unix prompt, you are logged in and ready to enter commands. However, before you start learning Unix commands, there are a few keys and key sequences you need to know about.

If you need to back up and correct a typing mistake in a command line, use the <DEL> key (known as the *erase character* in Unix). The <DEL> key is in the upper right-hand corner of the keyboard on all workstations. On RT workstations, it's labeled ''Backspace''; on VAX workstations, it's labeled with an X inside a back-arrow. Keep hitting this key until you've erased your mistake, and then retype the command correctly.

The **<CTRL>-c** sequence produces the Unix *interrupt character*.[1] If you have entered a command that you want to get out of, try entering <CTRL>-c. If one doesn't work, try several. Unfortunately, some programs ignore <CTRL>-c and refuse to quit. Try entering **q** (for **q**uit), **exit**, or **<RET> <CTRL>-d**. If you get really stuck, see a consultant.

---

[1] *<CTRL>-c* is one of several useful **control characters**. Any key sequence that begins with the <CTRL> character produces a *control character*. To type a *control character*, use the <CTRL> as you would the <SHIFT> key, then type the rest of the sequence.

3.4.1 Initialization Files

Initialization files are the files in your *home directory* that set up the workstation's environment when you login. As a new user, you have been given a copy of Athena's standard initialization files. You can list them with your other files by typing **ls -a**. Their names are *.login*, *.cshrc*, and *.mh_profile*.

You may eventually want to customize these files to change certain aspects of your workstation's environment. It is a good idea to be conservative when you do so; otherwise, you could jeopardize the integrity of your locker. If you ever get things hopelessly confused, you can always get another copy of these files from the directory */usr/prototype_user*. See Section 7 in Appendix C of this document for more information about customizing these files.

3.4.2 Changing Your Password: *passwd*

To protect your work and privacy, you should change your password from time to time. To change your password, use the *passwd* command. Type **passwd** and press **Return** or **Enter**; *passwd* will ask you for your old password, and then ask you twice for your new password. Neither your old nor your new password will appear on the screen when you type:

```
athena% passwd
Old password:                          <--- enter it invisibly
New Password for username:             <--- enter it invisibly
New Password for username:             <--- enter it again
Request was successful.
athena%
```

If you decide not to change it, type <CTRL>-c before finishing.

3.4.3 More Details

When you login to a workstation, the *login* process normally connects itself to your locker and places you in its directory. The name of this directory is **/mit/*username*** (where *username* is your username). This is your **Home Directory**.

If, for some reason, the workstation can't connect to your *NFS* locker, the *login* process displays the following warning:

```
Warning: Unable to attach home directory.

NOTE -- Your home directory is temporary.
It will be deleted when this workstation deactivates.


Your home directory cannot be attached right now.  You may use a
temporary directory, but

                       **** WARNING ****

If you do, any files you make or any mail you incorporate will be
deleted when you logout and lost forever.

Would you like to continue this login session, using a TEMPORARY
directory? (yes,no) [The default is no]
```

When this happens, you do not have access to your locker, so you can't get to your permanent files. Any work you do on the workstation, *including accessing your mail*, will be deleted after you logout, so if you want to save anything, you need to bring the appropriate removable media (floppy disks or tape cartridges). See Chapter 5 of this document for more information about removable media.

4. Files and Directories

If you are going to use the Athena system to do any substantial work, you are going to create and manipulate files. Writing a paper? You'll store it in a file. A program? A file.

A **file** is simply a bunch of data associated with a name (the *filename*). Most of the things you do in Unix affect files or their contents. Even the commands you type are filenames.

Actually, there are two main types of files, **simple files** and **directories**. Simple files contain text. If you're writing a paper, the file contains English text; if you're writing a C program, it contains C language text. If you're compiling a program, the compiler creates a file containing *binary* text. Most of the time you will use a text editor such as *emacs* to create and edit simple files.

Directories provide a means of organizing the simple files into meaningful groups. If you plan to write a bunch of papers during the term, you may want to keep them together in a directory called *Papers*. If you're into programming, you may want to store your programs in a directory called *Programs*, and within that directory, store the C programs in a **subdirectory** called *C_progs*, the Fortran programs in a subdirectory called *F_progs*.

4.1 Directories, Pathnames, and the Unix File System

The Unix file system is just a bunch of simple files organized into a *tree* of directories and subdirectories. Every directory and every file has a **pathname** that specifies its location within the file system tree.

For instance, when you login to an Athena workstation, the system puts you into a **home directory** that has the **pathname** */mit/username* (where *username* is your username). The home directory's name is *username*. It is a subdirectory of the directory *mit*, which in turn is a subdirectory of the file system's **root directory**. The *root directory* is named **/** (slash).

Schematically, the */mit/username* pathname looks like this:

```
   / (root directory)
   |
   |
  mit
   |
   |
username
```

And it's just one branch in a large tree that looks something like this:

```
                          / (root directory)
                          |
                          |
    ---------------------------------------------------
    |            |            |            |            |
    |            |            |            |            |
   bin          dev          etc    ...   mit    ...   usr
                              |            |            |
                              |            |            |
                         -----------       |       ------------
                         |         |       |       |          |
                        arp  ...  athena   |     athena ...  beta
                                           |
                        ---------------------------------------
                        |            |            |            |
                        |            |            |            |
                    username     username2     course1      course2
```

The diagram above is a gross simplification of the tree. Each of the *root level directories* (*bin*, *etc*, *usr*) have their own subdirectories and files, which in turn have their own subdirectories and files, and so on. Later in this chapter, the section entitled *File Organization: An Aside* briefly describes the contents of the system's more important directories.

4.2 Listing a Directory's Contents with the ls Command

The *ls* command lists the files contained in a directory. Suppose you are in your home directory. An *ls* command might yield something like the following:

```
athena% ls
Mail   welcome
athena%
```

These are two items that will be automatically placed in your locker. Actually, the directory contains other files, as you'll see in a minute.

4.3 The Dot Files .login and .cshrc

Mail and *welcome* aren't the only files that come with your home directory, but *ls* won't list the others because they are *dot files*, files with names that are prefixed with a dot **.** (a period). Use the **-a** option of *ls* to list them (*a* for *a*ll files):

```
athena% ls -a
.             .cshrc        .mh_profile
..            .login        Mail          welcome
```

Here's a quick explanation about the dot files listed above:

- The file **.** is the current directory, in this case your home directory */mit/username*. The file **..** is the current directory's parent directory, in this case your home directory's parent */mit*. All Unix directories refer to themselves as **.** and to their parent directory as **..**.

- The *.login* and *.cshrc* files are the *initialization files*. They help establish your working environment at login time. For more information, see the document ***More Unix: the C Shell***.
- The *.mh_profile* file sets the proper environment for electronic mail. For more information about this file, see the document *Essential Messages*.

4.4 Naming Directories and Files, Filename Extensions

Directory and filenames can be up to 256 characters in length, for what it's worth. You can use any character on the keyboard in a filename except a slash (/), but it is best to stick to a-z, A-Z, 0-9, the period or ''dot'' (.), and the underscore (_). If you use any other characters, you may be setting yourself up for serious trouble. Case matters in filenames: *myfile* is a different file from *Myfile*.

Unix filenames often have *extensions*. For example, in a file named *foo.txt*, the ''*.txt*'' is the extension. Files have extensions for two reasons. First, because an extension tells people something about a file; for example, a *.h* file is a program header file, and a *.mss* file is a *scribe manuscript* file. Second, because some Unix programs require that the files they deal with have a certain extension; for example, the C compiler requires that its source files have a *.c* extension.

4.5 Wildcards

Unix provides a mechanism called *wildcard characters* that lets you refer to more than one file at once. Here are the two most commonly used wildcard characters:

*	Matches all files except dot files. It can also match characters within a filename. For example, **.f* means ''all filenames with a *.f* extension''; *h.f* and *verylongfilename.f* would both match. *a** means ''all filenames that begin with *a*''; *a* and *anotherfilename* would both match.
?	Matches single characters. *?.f* means ''all one character filenames with a *.f* extension''; *a.f* would match but *ab.f* would not.

4.6 Working With Files

The following commands let you list, examine, create, delete, copy, and rename files.

4.6.1 Other Options to the ls Command

The *ls* command has many options. To see them all, use the *man* command to look at the online manual page for *ls* by typing **man ls**. Here are some of the more useful ones.

-l for Long

To get a list of your files that shows more information about them, type **ls -l**.

```
% ls -l
total 2
-rw-r--r--  1 username        1081 Jul 30 17:00 welcome
%
```

The first item on the line is the file's *mode*. A ''d'' in the mode's first column means that the file is a directory. The r's, x's, and w's in the mode show who has permission to read, write, delete, and execute the file. The second item on the line is the number of *links* the file has. We won't explain links here, but for directories, it shows how many subdirectories exist beneath the file; there are always at least two. The third item is the username of the user who owns the file. In most cases this is your username. The fourth item is the size of the file in bytes. For a text file, this is the number of characters in the file. The fifth item is the date and time when the file was last modified. If a file has never been modified, it shows the date it was created. The sixth and final item is the filename.

-F for Type

As we said before, there are different *types* of files: simple files, which can be *text files*, *binary files*, or *shell scripts*; and directory files. Often you need to know the type of the files you're listing. The *-F* option does this:

```
% ls -F
Mail/    a.out*   myfortpgm.f   welcome
%
```

Notice the suffix characters (/ and *) following some of the filenames. These characters are NOT part of the filename. The / suffix means that the file is a *directory file*. The * suffix means that the file is a *binary file* or a *shell script*. No suffix means either that the file is a text file or Unix cannot recognize its type.

You can combine *ls* options to use more than one at the same time. For example, to get a *long listing* of *all the files* in a directory, type **ls -*la***.

4.6.2 Seeing the Contents of a File

There are two commands that display the contents of a file: *cat* and *more*. Although the *cat* command is simpler, most people prefer the *more* command.

cat

The *cat* command (for **cat**enate) displays the contents of a file on the screen. Its command format is **cat *filename***.

If the file is very long, its contents will scroll past you faster than you can read them. You can type **Ctrl-s** to stop the screen from scrolling and **Ctrl-q** to resume scrolling. However, if you have to do this, you should probably be using the **more** command, which doesn't let this happen. To cancel a *cat* command and stop its output, type **Ctrl-c**.

NOTE: The workstation may print out garbage and beep a lot after you ask it to ‘‘cat’’ certain files. This is because some kinds of files are unprintable; they contain data in a format that cannot be displayed on the screen. Binary files (the ones *ls -F* shows with a * suffix) are not printable; neither are directory files (the ones *ls -F* shows with a / suffix).

more

The *more* command displays a file’s contents one screen at a time, pausing after every screenful so that you can read them. Its command format is **more *filename***.

Once *more* has filled up the screen, you must give it a *subcommand* to tell it to continue. There are many such subcommands. Here are the most commonly used ones:

SPACEBAR	Move forward one screenful
RETURN	Move forward one line
b	Move back one page
***/word*RETURN**	Search forward for ***word***
q or **Ctrl-C**	Quit

More is clever enough not to display undisplayable files.

More is what is called a Unix *filter*, that is, many other Unix commands filter their output through *more*: instead of giving you their output directly, they send it through *more* so that it will be easy to read and under your control. In Unix parlance, we say that they ‘‘*pipe* their output through *more*.’’ The *man* command that displays online documentation does this.

You, too, can ‘‘pipe things through *more*.’’ Whenever you enter a command and its output is too large to fit on one screen, cancel it by typing **Ctrl-c** and re-enter it like this:

`athena%` ***command*** **| more**

The ‘‘|’’ is the *pipe* character. For example, if you have collected hundreds of files, you can pipe the output of *ls* through *more* by typing **ls -l | more**.

Type **man more** to learn more about *more*.

4.6.3 Creating a File

There are many ways to create a file. The usual way to do so is with the *emacs* text editor. See the Athena document ***Essential Emacs*** for more information.

A quick and easy way to create a file is with the *cat* command. It works as follows:

```
athena% cat > easyfile
Unix moves the cursor down to the next line.  Start typing.
You can fix typing mistakes in the current line with the
DELETE key.  You can erase the current line by typing Ctrl-u.
You cannot change previous lines.  When you are
done, finish the last line with a RETURN and then type Ctrl-d.
It echoes as ^D.
^D
athena%
```

The > in the command line is the Unix *redirection mechanism*. It tells the *cat* command to *redirect* its output somewhere other than the screen (in this case, into *easyfile*). With no arguments, the *cat* command simply echoes what you type, so it's literally echoing your input into a file. Be sure the arrow is pointing the right way. It is all right to have a blank between the > and the filename.

You can redirect the output of any Unix command into a file. For example, you can put a listing of your files into a file so you can print it out and refer to it at your leisure:

```
athena% ls -l >mylisting
athena% ls
easyfile        mylisting       welcome
```

4.6.4 Deleting a File with the rm Command

It is a good idea to delete files that you know you won't need anymore. Otherwise, your directory can become very full. The command to delete a file is *rm* (for **rem**ove). Its command format is **rm** ***filename***. Unix doesn't verify the deletion; it just prompts for the next command. You can put more than one filename on the command line. You can use wildcards (with caution!) to delete groups of files. The most catastrophic mistake you can make is to type **rm ***, which deletes *all* the files in the current working directory. Or worse yet **rm -r ***.

Note: It is a **very** good idea to use **rm -i** to delete files. The *-i* option stands for *interactive*, which will ask you to confirm the deletion you are performing. For example:

```
athena% rm -i myfile.txt
rm: remove myfile.txt? n
athena% rm -i yourfile.txt
rm: remove yourfile.txt? y
athena%
```

4.6.5 Moving or Renaming a File with the mv Command

To move or rename a file, use the *mv* command. Its command format is **mv** ***fromfile tofile***. Unix renames ***fromfile*** as ***tofile***, in effect moving it from one file to another. If ***tofile*** already exists, it is overwritten. As ever, Unix does not verify that it has moved the file.

4.6.6 Copying a File with the cp Command

The *cp* command copies a file. Its command format is **cp *fromfile tofile***. Unix makes a copy of ***fromfile*** called ***tofile***. If ***tofile*** already exists, it is overwritten. Again, Unix does not verify the copy.

4.6.7 The -r Recursive Option

Many commands have a *-r* or *-R recursive* option. When referring to directories, recursive means ''do it to this directory and to all the files and subdirectories underneath it.'' Recursive options are thus very powerful. For example, you can copy an entire subtree into your home directory by typing:

athena% cp -r /foo/bar ~*username*

The *rm* command has a recursive option which makes it dangerously easy to delete an entire subtree. The *rmdir* command does not have a recursive option.

4.7 The Current Working Directory

Your *current working directory* is the directory you are ''in'' at any particular time. From the time you login to the time you logout, you are in a current working directory. You start out in your home directory because that is where the *login* process puts you when it gives you control of the workstation.

4.7.1 The pwd Command

The *pwd* command displays the pathname of your current working directory (*pwd* stands for **p**rint **w**orking **d**irectory). If you use the command immediately after logging in, the transaction will proceed as follows:

athena% pwd
/mit/*username*
athena%

where *username* is your username.

4.7.2 Changing Directories with the cd Command

The *cd* command stands for **c***hange working* **d***irectories*. Suppose you want to change your working directory from your home directory to the directory above it, *mit* (*mit* is your home directory's *parent*). Enter the following commands:

```
athena% pwd
/mit/username
athena% cd /mit                 <-- Change working directory.
athena% pwd                     <-- Print new directory's path.
/mit
athena% ls                      <-- List its contents.
username
athena%   cd /mit/username      <-- Change back.
athena%   ls
Mail      welcome
```

If you are in a directory other than your home directory, you can use *cd* without an argument to change the working directory back to your home directory:

```
athena% pwd
/mit/username
athena% cd /mit
athena% pwd
/mit
athena% cd                      <-- Change back to home directory.
athena% pwd
/mit/username
athena%
```

If you don't specify a directory when using *cd*, the command will always return you to your home directory.

4.8 Creating Directories

You use the *mkdir* command to create directories. Suppose you are in your home directory and you want to create a series of directories in which to store your programs. You want a directory called *Programs* with the subdirectories *C_progs* and *F_progs*. The following series of commands does the job and checks it along the way:

```
athena% pwd
/mit/username                   <-- From your home directory . . .
athena% mkdir Programs          <-- Make the subdirectory.
athena% cd Programs
athena% pwd
/mit/username/Programs          <-- From Programs directory . . .
athena% mkdir C_progs           <-- Make C_progs subdirectory,
athena% ls
C_progs
athena% mkdir F_progs           <-- and F_progs subdirectory.
athena% ls
C_progs      F_progs
athena%   cd                    <-- change back to your home directory.
athena% pwd
/mit/username
```

4.9 Removing Subdirectories

If you want to remove a directory, you will use one of two commands: **rmdir**, or **rm -ir**. The *rmdir* command removes **empty** directories, and *only* empty directories. If a directory isn't empty, *rmdir* will display an error message. You must then *cd* to that directory and remove all of its files and subdirectories.

The *rm -ir* command (actually it's the *rm* command with the *-r* option and with the *-i* option) *recursively* removes directory contents--its files and subdirectories--then the directory itself. If any of the directory's subdirectories have contents, those are removed before the subdirectory. *rm -r* is a powerful, and hence, **dangerous** command. Make sure you know what you're doing when you use it; the *-i* option is a helpful safety.

The following session illustrates the use of both commands:

```
athena% pwd                  <-- From your home directory . . .
/mit/username
athena% rmdir Programs       <-- Try to remove the Programs directory.
rmdir: Programs: Directory not empty
athena% cd Programs
athena% ls                   <-- cd to Programs and see what's there.
C_progs  F_progs
athena% ls F_progs           <-- What's in F_progs?  Nothing.
athena% rmdir F_progs        <-- You can rmdir to remove F_progs.
athena% cd
athena% rmdir Programs       <-- cd to home and try to remove Programs.
rmdir: Programs: Directory not empty
athena% ls Programs          <-- It's still not empty.
C_progs
athena% rm -r Programs       <-- All right, let's quit goofing around.
athena% ls Programs
Programs not found           <-- It's gone now.
athena%
```

Use *rmdir* and *rm -r* judiciously. It's an old Unix adage that says ***rm* is forever**.

4.10 Directory Abbreviations

With the exception of the *root directory* **/**, all directories have a parent directory. Unix provides an abbreviation for a directory's parent; it's the **..** directory. Here's another example:

```
athena% pwd                    <-- From your home directory . . .
/mit/username
athena% cd ..                  <-- cd to its parent directory.
athena% pwd
/mit
athena% cd ..                  <-- Now cd to mit's parent,
athena% pwd
/                              <-- which happens to be the root directory.
athena% cd                     <-- Then cd back to your home directory.
athena% pwd
/mit/username
```

4.10.1 The Directory ~ . . .

Is an abbreviation for your home directory. In other words, **cd ~** is equivalent to **cd**. Another for-instance:

```
athena% pwd
/mit/username
athena% cd /etc                <-- From your home directory . . .
athena% pwd                    <-- cd to the root-level directory etc.
/etc
athena% cd ~/Programs          <-- Now back to your subdirectory Programs
athena% pwd
/mit/username/Programs
```

4.11 Your File System Locker

Most of the time, when you login to a public workstation, that workstation creates your home directory and *attaches* your locker to it. Your locker is your home directory, and when you use the *ls* command to list its contents, you're listing a directory that is ''somewhere on the network'' on your file server machine.

If your file server machine is down or unreachable when you try to login, the workstation will be unable to *attach* your locker. When this happens, the *login* process displays the following message:

```
Warning: Unable to attach home directory.

NOTE -- Your home directory is temporary.
It will be deleted when this workstation deactivates.

Your home directory cannot be attached right now.  You may use a
temporary directory, but

                    **** WARNING ****

If you do, any files you make or any mail you incorporate will be
deleted when you logout and lost forever.

Would you like to continue this login session, using a TEMPORARY
directory? (yes,no) [The default is no]
```

And of course you don't have access to your permanent home directory. Press **Return** or **Enter** to halt the *login* process, and try logging in later.

What happens if the file server becomes unreachable while you're in the middle of a login session? You temporarily lose access to your files. **You don't lose anything permanently.**

You can tell that the file server is gone if you use any of the directory commands while in your home directory or any of its subdirectories. Suppose you're in your home directory and you try to use *ls*. If something is wrong, your workstation will begin displaying **black bar** messages that say something like this:

```
athena% ls
NFS getattr failed for server server: TIMED OUT
```

This message may appear a thousand times while you watch in disbelief.

If you stay logged in and wait long enough, your files will reappear when the network heals. If you get tired of waiting, make sure you logout. Otherwise the next person who uses the workstation will have access to your files when the network comes back.

4.11.1 But I'm in the Middle of an Emacs Session!

Don't panic. Use the *emacs* command **Ctrl-x Ctrl-w**, and write the file in the directory **/usr/tmp**. The file will stay in that directory on that particular workstation for four days, so you have ample time to move it to your locker when the network comes back.

4.12 File Organization: An Aside

This chapter has mainly talked about your own files. But, as anyone new to Boston must, over time, construct a mental map of the city, its suburbs, squares, T stops, restaurants, movie theatres, and points of personal interest beyond their immediate neighborhood, so should you get an inkling of what is kept in other parts of the Unix file system. Here's a brief tour.

Type **ls -F /**. You will see something like:

```
bin/       flp0/         mnt/     ultrixboot
boot*      flp1/         site/    urvd/
bootblock  lib/          srvd/    usr/
dev/       lost+found/   stf/     vmb.exe*
etc/       mit/          tmp/     vmunix*
```

Each of these *root level directories* (the ones followed by a /) contains different sorts of files that Unix needs in order to function. There are strong historical conventions for what is kept in each of them (as well as many arbitrary exceptions). You can look at most of the files in each of these directories. With a few exceptions, Unix is open for inspection by anyone curious enough to look. Here is a short description of what's kept in some of these directories:

The */bin* directory contains the binary (unreadable, executable machine language) versions of the programs you absolutely can't bring up Unix without, such as the *cat* and *rm* commands. When you type the *cat* command, you are actually executing the file */bin/cat* in this directory.

The */etc* directory contains all the administrative information files necessary to run the system. This includes the */etc/passwd* file that defines who has a username on the system. It makes for interesting browsing.

The */mit* directory, as you already know, contains your files and the remote software packs.

Perhaps the most interesting directory is */usr*. */usr* has traditionally been where users' files were kept. (Athena uses */mit* for this.) Later, it also came to be the place where documentation, games, and programs added at the local site were kept. For example, */usr/ucb* has all the programs that the University of California at Berkeley added to the original Unix to make Berkeley 4.0. */usr/athena* has all the programs, like Scribe and Emacs, that Athena is adding to the Unix it got from Berkeley. The */usr* directory is worth quite a bit of browsing.

The *Student Information Processing Board* (*sipb*) maintains a file system full of interesting software packages that you can *attach* to your workstation after you have logged in. After you have *attached* this file system to your workstation, you can *cd* to the */usr/sipb* directory and look around. See Section 3 of Appendix B for more information about *attaching* the *sipb* file system.

5. The X Window System

The X Window System is a group of programs that let you divide your workstation's screen into one or more overlapping rectangular **windows**. Each window acts like a separate terminal, and you can do different things in each one. You can write a paper with *emacs* in one window, format a letter with *scribe* in another, compile a program in another, view your mail messages in still another, and so on.

X makes the workstation a very powerful instrument. After you create the windows you want to use, you can move them around, shuffle them above or beneath one another, and make them any size you want. And you can remove individual windows from your screen at your pleasure. You manipulate windows using certain keyboard and mouse-action combinations.

5.1 Of Mice and Windows

The *mouse* is the small puck with some buttons on it that is attached to your workstation. The mouse controls the *mouse cursor* on your screen.

When the mouse cursor is inside a window, it looks like a capital **I**, and when it is over the *background screen* (the area outside every window), it looks like a hollow capital **X**. When the mouse cursor is inside a window, that window is called the *active window*. Whatever you type at the keyboard appears in the active window. When the mouse cursor is over the background screen (also called the *root window*), typing at the keyboard has no effect.

Although X is the same on both the VAX and RT workstations, there are two differences between the two brands that affect you when you use X. The VAX mouse is different from the RT mouse. The VAX mouse has three buttons and the RT mouse has two.

On the RT mouse, *pressing both buttons at the same time is equivalent to pressing the middle button on the VAX mouse*.

Another difference is in the location of the *META key*. On the VAX, the META key is labeled "Compose Character"; it's located just to the left of the space bar. On the RT, the META key is labeled "Alt"; it's located on either side of the space bar.

There are three actions you can perform with the mouse: *pointing*, *clicking*, and *dragging*. *Pointing* is simply moving the mouse to put the mouse cursor in the right place. *Clicking* is quickly pressing and releasing any one of the mouse buttons (or two, for the "middle button" on the RT). *Dragging* is pressing a mouse button and holding it down while simultaneously moving the mouse.

5.2 Creating Windows with the xterm Terminal Emulator

The *xterm* program lets you create an additional window on your screen that acts like a separate terminal, hence its name.

5.2.1 Creating a Window

To create a new window, just type:

`athena%` **xterm &**

Unix types back:

```
[1] 8389
athena%
```

Using the "**&**" causes the *xterm* program to run as a *background process*. If you don't type the **&**, you won't be able to work in your original window, since you will be running *xterm* in it. The [1] means that this is your first background process and the 8389 is its Process Identification number, or PID.

After you type **xterm &**, the mouse cursor will change into a bracket with a tiny "rubber band" window inside it. Also, in the upper left-hand corner of the screen, a small box will appear that says "xterm: 0x0". The 0x0 is the dimensions of the rubber band window in columns by lines. Now move the bracket (by moving the mouse) to the location on the screen where you want the upper left-hand corner of the window to be. With the three mouse buttons, there are three things you can do:

- **Click the left button** to create a window that is 80 columns wide and 24 lines high.
- **Click the right button** to create a window that is 80 columns wide and extends to the bottom of the display screen.
- **Drag on the middle button** to create a window of any size. As you move the mouse, the numbers in the box in the upper left-hand corner will change appropriately. Just release the middle button when the window is the size you want. **To perform this function on the RT, you must press both mouse buttons and drag.**

After you have created a window, you have to wait a few seconds for the prompt to appear before you can type in it. Once the prompt appears, you've got another window to work in!

5.2.2 Getting Rid of a Window

You can get rid of a window by typing **exit** in it. This **kills** the *xterm* process and makes the window vanish.

You do not have to get rid of all of the windows that you create before you logout, though it is generally safer (except for *emacs* windows--exit these normally); the logout process will get rid of them for you. However, *you can logout only from your original login window*. If you try to logout from any other window, you will get the message "Not login shell.".

The *xterm* emulator has a million features that let you customize the size, position, and font type of your windows. For more information, see ***Appendix C*** of this document.

5.3 The Window Manager

When you login, the *login* process starts a **u**nix **w**indow **m**anager (**uwm**) for you. The window manager lets you shuffle, move, and resize the windows on your screen. Once the window manager has started, you don't need to start another.

5.3.1 Using the Window Manager

There are two ways to issue window manager commands: you can use the mouse, or you can choose commands from a pull-down menu.

You issue commands from the mouse with various mouse button clicks and dragging combinations. **But the window manager does not recognize these combinations unless you press and hold the workstation's meta key at the same time.**

Since there are only three buttons on the mouse (two on the RT mouse, but pressing them at the same time gives you a middle button), and two ways of using each (clicking and dragging), you can use the mouse to issue six different commands.

***uwm* Mouse Commands**

	Left Button	**Middle Button**	**Right Button**
Click	Push a window beneath others	Restore iconified window	Bring a window to the front
Drag	Iconify/deiconify a window	Resize a window	Move a window

Here is a description of each of these commands:

Pushing a window beneath others
Think of your screen as displaying a *stack* (a messy pile) of windows. Clicking the left mouse button will push the active window to the bottom of the stack, away from you.

Bringing a window in front of others
Clicking the right mouse button will bring the active window to the top of the stack.

Moving a window
Hold the right mouse button down. As soon as you move the mouse, the active window will be outlined with a "rubber band." Then move the window to its new location by moving the mouse.

Resizing a window
To change the size of a window, move the mouse cursor to the corner of the window you want to resize. Hold the middle mouse button down; the mouse cursor will change into a cross with arrowheads on the ends. Move the mouse until the window is the desired size. Notice that in the corner opposite the mouse cursor, a small box displays the window's height and width in columns by lines (e.g., 80x24). As you change the size of the window, the numbers in the box change accordingly. Release the button when

the window is the size you want.

NOTE: After you resize a window, programs that use the window size to help them display their output will automatically know the correct size of the window. Perhaps one of these programs might not believe that the window is the size it is. If this happens, type **xresize** inside the window. (*xresize* is an *alias* that was defined in the system-wide *.cshrc* file.) If you have any questions about this, read the online manual page on *resize* by typing **man resize** or ask a consultant.

Iconifying a window An *icon* is a small symbol that represents a window. It remembers the size and location of that window, so when you *restore* the iconified window, it will appear in its old location. You should iconify windows to clean up your screen if there are too many for you to easily see them all.

To iconify a window, hold down the left mouse button and move the mouse; the mouse cursor will change into a cross with arrows on the ends. Now move the mouse cursor to where you want to put the icon, and release the button. The window will vanish, and its icon will appear at the mouse cursor's location.

You can move icons just like window. Place the mouse cursor on the icon, press the right mouse button, and drag.

NOTE: You can change an icon's title if you don't like it. To do so, move the mouse cursor over the icon, hit the DELETE key to remove the existing title, and then type in the new title. (Some icons don't have titles, but *xterm* icons do.)

Restoring an iconified window

To restore an iconified window, move the mouse cursor over the icon and click the middle mouse button. The window will reappear in its old location.

5.3.2 Window Manager Commands From a Menu

You can also issue window manager commands by making selections from a pull-down menu. To produce this menu, hold down the SHIFT key, the META key, and the middle mouse button. (The mouse cursor can be anywhere on the screen.) A menu called **WindowOps** will appear.

Drag the mouse cursor (which now looks like an arrow) down the menu to the command you want, and then release the buttons. *Now*, move the mouse cursor into the window you want the command to affect, and press the META key and the middle mouse button. The command will take effect. You may need to drag the mouse around, depending on which command you select.

6. Backing Up Your Files to Removable Media

Backing up files means making copies of them to cover yourself in case you inadvertantly delete something important.

If the disk drive that hold your locker fails, Project Athena will attempt to restore your files from backup. We cannot restore anything that you accidentally delete with the *rm* command. If you **really** want to protect your term paper or your thesis, you must back up your important files to removable media.

What's "removable media"? At Project Athena, removable media comes in the form of floppy disks and tape cartridges. Removable media can be purchased at the Information Services Microcomputer store in room 11-209. The store is usually open between the hours of 10 p.m. and 4 p.m. on weekdays and prices can range from about $2.00 for disks to $25.00 for tape cartridges.

If you are going to back up your files with removable media, plan to set up a backup cycle with a rotation of two or more backup sets:

1. Back up your files to floppy A;
2. Next time, back up your files to floppy B;
3. Next time, back up your files to floppy A (or floppy C);
4. And so on.

If you keep multiple copies of your backups, you won't rip your hair out if you accidentally mangle one of them.

6.1 Types of Removable Media

Project Athena workstations come in four flavors: the DEC VS2000, the IBM RT, the DEC VSII, and the DEC VSII/RC. All workstations have removable media devices. The DEC VS2000 and IBM RT have floppy disk drives; some DEC VSIIs have dual floppy disk drives, others have tape drives; some DEC VSII/RCs have dual floppy drives, others have tape drives.

Because the disk drives on the VS2000 and IBM RT can both use *1.2M double-sided* **high** *density* floppies, we recommend that you back up your files using 1.2M diskettes. RT drives are generally more reliable than those on the VS2000. It is also important to note that disk formatting on the VSII is incompatible with the RTs and VS2000s. The instructions that follow are for the VS2000 and the RT workstations.

6.2 Formatting 1.2M Double-Sided High Density Diskettes

1.2M Double-sided high density diskettes must be **formatted** before they can be used for backup purposes. When you format a diskette, you remove any data that may have been stored on it. Obviously this is not a big deal if you are formatting a new diskette, but if you are formatting a used one, make sure you know what you are doing.

To format a floppy on an RT workstation, login, insert the floppy into the disk drive, and enter the following command:

```
athena% /etc/fdformat -h /dev/rfd0
```

The system prompts you with the following message:

```
format /dev/rfd0 to 1.2Meg floppy?
```

Enter a **y** to proceed with the format. Enter any other letter to halt the procedure.

6.3 Backing Up and Restoring Files with the tar Command

The command named **tar** (the antiquated **t**ape **ar**chive procedure) is a complicated one; it has more options than you can shake a stick at. In fact you will probably want to hit it with a stick when you try to use it.

This section will tell you how to back up and restore files by rote. Don't expect to understand the various command incantations. Just follow the examples and do exactly the same thing when you back up your own files.

If you want to find out more about *tar*, read the manual page and experiment. To see the manual page, type **man tar**.

6.3.1 The Example

Pretend you have a directory within your home directory called *Papers*. It contains the papers you are writing for this term's courses. When you list its contents with *ls*, here's what you get:

```
athena% ls Papers
Cowboys         aristotle.mss  madd1.mss
Shakespeare     edgar_poe.mss  p_glass.mss
athena%
```

6.3.2 Will it all Fit?

An important question. The data you want to save must be able to fit on the floppy diskette which can hold up to **One Megabyte** of information. (The diskette is supposed to hold 1.2 Meg, but experience shows that 1 Meg is a better bet.)

If you try to stuff too much data onto the diskette, *tar* complains as follows:

```
tar: tape write error: No space left on device
```

When this happens, you cannot reliably recover anything that *tar* copied to the floppy. But don't worry. The diskette isn't ruined. You can make another backup by recopying over the existing data.

You can use the **du -s** command to determine the amount of data you are trying to save. This command measures a directory's size in terms of the number of **blocks** it contains (*1 block = 1Kbyte ; 1024K = 1Meg*). If a directory's contents consume more that 1000 blocks of storage, the contents won't fit on a single diskette--you must subdivide to other directories.

6.3.3 Backing Up the Papers Directory

The backup procedure involves three steps:

1. You use the *cd* command to change directories to the directory you want to save;
2. you use the *du -s* command to determine the size of the data you are trying to save (you have to make sure it will all fit on one floppy diskette); and
3. you use the *tar* command to save the files.

Assuming that you have placed a **formatted** 1.2M high density floppy diskette into the appropriate place, execute the following commands:

```
athena% cd Papers               <-- From your home directory, cd to Papers.
athena% pwd
/mit/username/Papers
athena% du -s                   <-- Determine number of blocks in Papers.
92      .                       <-- Less than 1000. It will fit.
athena% tar cvf /dev/floppy .   <-- The dot "." is important!
                                    It represents your current directory.
a ./p_glass.mss 1 blocks
a ./Shakespeare/notes 1 blocks
a ./Shakespeare/beginning.mss 6 blocks
a ./Shakespeare/part2.mss 5 blocks
a ./Shakespeare/newbeg.mss 8 blocks
a ./aristotle.mss 21 blocks
a ./Cowboys/blues 2 blocks
a ./Cowboys/cactus 1 blocks
a ./Cowboys/cowboy 2 blocks
a ./Cowboys/notes 3 blocks
a ./Cowboys/s1v2 10 blocks
a ./Cowboys/s1v3 12 blocks
a ./Cowboys/scene1 8 blocks
a ./Cowboys/scene1notes 1 blocks
a ./Cowboys/tequila 1 blocks
a ./madd1.mss 2 blocks
a ./edgar_poe.mss 86 blocks
athena%                         <-- tar has copied the files to the floppy.
```

A note about *tar* and the **c** option: *tar cvf* **will destroy any information that exists on the floppy** (**c** stands for **c***reate new archive*). Don't use *tar cvf* on a floppy that already contains files unless you are sure you want to overwrite the floppy.

6.3.4 Listing the Contents of a Floppy

Want to see what a floppy contains? Insert the floppy in the drive and use the **t** option to *tar* as follows:

```
athena% tar tvf /dev/floppy
rwx--x--x10476/101      0 Sep  8 08:40 1988 ./
rw-------10476/101    186 Sep  8 07:10 1988 ./p_glass.mss
rwx--x--x10476/101      0 Sep  8 07:10 1988 ./Shakespeare/
rw-------10476/101    137 Sep  8 07:10 1988 ./Shakespeare/notes
rw-------10476/101   2774 Sep  8 07:10 1988 ./Shakespeare/beginning.mss
rw-------10476/101   2381 Sep  8 07:10 1988 ./Shakespeare/part2.mss
rw-------10476/101   3971 Sep  8 07:10 1988 ./Shakespeare/newbeg.mss
rw-------10476/101  10453 Sep  8 07:10 1988 ./aristotle.mss
rwx--x--x10476/101      0 Sep  8 07:10 1988 ./Cowboys/
rw-------10476/101    709 Sep  8 07:10 1988 ./Cowboys/blues
rw-------10476/101    274 Sep  8 07:10 1988 ./Cowboys/cactus
rw-------10476/101    925 Sep  8 07:10 1988 ./Cowboys/cowboy
rw-------10476/101   1360 Sep  8 07:10 1988 ./Cowboys/notes
rw-------10476/101   4917 Sep  8 07:10 1988 ./Cowboys/s1v2
rw-------10476/101   5841 Sep  8 07:10 1988 ./Cowboys/s1v3
rw-------10476/101   4018 Sep  8 07:10 1988 ./Cowboys/scene1
rw-------10476/101     95 Sep  8 07:10 1988 ./Cowboys/scene1notes
rw-------10476/101    385 Sep  8 07:10 1988 ./Cowboys/tequila
rw-------10476/101    675 Sep  8 07:10 1988 ./madd1.mss
rw-------10476/101  43932 Sep  8 07:10 1988 ./edgar_poe.mss
athena%
```

6.3.5 Restoring the Entire Contents of the Floppy

Suppose you want to restore the entire Papers directory from the floppy. Use the **x** option of *tar* (*x* stands for *e***x***tract*), and do something like the following:

```
athena% mkdir Restoredir          <-- From your home directory . . .
athena% cd Restoredir             <-- Create and cd to a Restoredir.
athena% tar xvf /dev/floppy       <-- Restore the files.
x ./p_glass.mss, 186 bytes, 1 tape blocks
x ./Shakespeare/notes, 137 bytes, 1 tape blocks
x ./Shakespeare/beginning.mss, 2774 bytes, 6 tape blocks
x ./Shakespeare/part2.mss, 2381 bytes, 5 tape blocks
x ./Shakespeare/newbeg.mss, 3971 bytes, 8 tape blocks
x ./aristotle.mss, 10453 bytes, 21 tape blocks
x ./Cowboys/blues, 709 bytes, 2 tape blocks
x ./Cowboys/cactus, 274 bytes, 1 tape blocks
x ./Cowboys/cowboy, 925 bytes, 2 tape blocks
x ./Cowboys/notes, 1360 bytes, 3 tape blocks
x ./Cowboys/s1v2, 4917 bytes, 10 tape blocks
x ./Cowboys/s1v3, 5841 bytes, 12 tape blocks
x ./Cowboys/scene1, 4018 bytes, 8 tape blocks
x ./Cowboys/scene1notes, 95 bytes, 1 tape blocks
x ./Cowboys/tequila, 385 bytes, 1 tape blocks
x ./madd1.mss, 675 bytes, 2 tape blocks
x ./edgar_poe.mss, 43932 bytes, 86 tape blocks
athena%
```

There is an alternative to creating the **Restoredir** if you are concerned about file space. You can cd in to **/tmp** (*Athena temporary directory*) and use that instead. Just make sure to check that you can read your files before you actually delete them.

6.3.6 Restoring Single Files

To restore a single file from a floppy, you have to specify it exactly as it is named on the floppy. For instance, suppose you want to restore the file *p_glass.mss*. When you listed it a s shown above, *tar* gave its name like this:

```
rw-------10476/101    186 Sep  8 07:10 1988 ./p_glass.mss
```

To restore the file, you must give its full name as follows:

```
athena% tar xvf /dev/floppy ./p_glass.mss
```

6.3.7 Backing Up a Series of Single Files

You can back up single files with *tar* without having to back up entire directories. Suppose that within the *Papers* directory you want to back up just the file *aristotle.mss*. Using a brand new *formatted* floppy (or one that you don't mind overwriting), proceed as follows:

```
athena% tar cvf /dev/floppy aristotle.mss
a aristotle.mss 21 blocks
```

Now list the floppy's contents:

```
athena% tar tvf /dev/floppy
rw-------10476/101  10453 Sep  8 07:10 1988 aristotle.mss
```

You can add additional files to the floppy by using the **r** option to *tar*. **Do not use the *c* option to *tar* or you will overwrite the file or files you have already saved.**
Suppose you want to add the file *edgar_poe.mss* to the floppy. Proceed as follows:

```
athena% tar rvf /dev/floppy edgar_poe.mss
a edgar_poe.mss 86 blocks
```

Now if you list the contents of the floppy, this is what you get:

```
athena% tar tvf /dev/floppy
rw-------10476/101  10453 Sep  8 07:10 1988 aristotle.mss
rw-------10476/101  43932 Sep  8 07:10 1988 edgar_poe.mss
```

You can restore the files as follows:

```
athena% tar xvf /dev/floppy aristotle.mss
x aristotle.mss, 10453 bytes, 21 tape blocks
athena% tar xvf /dev/floppy edgar_poe.mss
x edgar_poe.mss, 43932 bytes, 86 tape blocks
```

A. File Protection

Each file in Unix has a **protection mode**. This includes text files, directories, and programs. The **protection mode** determines who may have access to a file, and what kind of access it is.

The protection mode is set with respect to three classes of users:

- **user**-- you
- **group**-- people associated with you through a research project or status as a student.
- **world**-- everyone

For each class you declare whether or not the file is:

- ***readable*** (**r**) -- the ability to look at or copy a file
- ***writeable*** (**w**) -- the ability to modify or delete
- ***executable*** (**x**) -- the ability to execute the file

For example, you might set the protection mode on *somefile* to be *readable*, *writeable*, and *executable* by the **owner**(yourself); while only *readable* and **executable** by **group** and **world**. This means that everyone may look at, copy and execute your file, but that only you may change it.

A.1 Determining the Protection on a File

The command **ls -l** includes the protection modes on the listed files. As shown below,

```
athena% ls -l

    total 4
    -rw-------  1 owner   210    Jan 21 18:12 file.mss
    drwx--x--x  2 owner   512    Jan 20 14:23 rs1.dir
    -rw-------  1 owner    45    Nov 14 14:51 setup
    -rw-------  1 owner    70    Jan  7 14:17 setup.repoman
    |________|
        |
      protection mode
```

the representation of the protections is a sequence of 10 symbols. The first item in the sequence is a special flag. Commonly it tells that the file is a directory (**d**), a link (**l**), or a normal file (**-**). The remaining 9 items correspond to the **read**, **write**, and **execute** permission of *owner*, *group* and *world* respectively. In other words, positions 2 through 4 are for **owner** *read*, *write*, and *execute* permissions. Positions 5 through 7 are for group permissions; and positions 8 through 10 are for **world** permissions. The appropriate symbol (**r,w,x**) in a given position implies that the permission exists, while the **-** symbol means that it does not. A directory that is only **owner** *readable* and *executable* would have the following representation:

```
dr-x------  1 owner    512     Aug 22 21:21 mydir
```

A file which was **owner** *readable*, *writeable*, and *executable*, as well as **group** and **world** *readable* and *executable* would look like this:

```
-rwxr-xr-x  1 owner     92     Aug 23 22:22 myfile
```

A.2 What is an Appropriate Protection Mode?

Access to files is restricted by the three protection characteristics. These must be properly set in order for you (and others) to have the access you desire.

Read permission allows a person to read and copy a file. Additionally, the reader ***must*** have execute permission on ***all*** of the directories between the file and his location.

Write permission allows modification of files. **Write** permission on a directory is required for the creation and deletion of files in that directory.

Execute access allows a person to run a file as a program. In the case of a directory this means the ability to **chdir** to it. You cannot look at lower level files unless the intervening directories have execute access.

An appropriately protected file requires that the modes be properly set on both the file of concern and all of the directories above it.

Your files have a default protection mode that lets only you read and write them. If you want to give others permission to read your files, you must take explicit action as described in the next section.

It is bad practice to set your protection mode to world writeable. This means that anyone may change, delete or harm your files.

A.3 Setting the Protection Mode

The command **chmod** (change mode) modifies a file's protection mode. The command **chmod** accepts either of two kinds of arguments. The argument may be a numerical value corresponding to the absolute value of the mode, or a symbolic representation of what to change relative to the current state.

A.4 Specifying the Protection Mode with a Numeric Value

The following table can be used to determine which numeric values correspond to which protection modes.

```
user      r..............400
          w..............200
          x..............100
group     r...............40
          w...............20
          x...............10
world     r................4
          w................2
          x................1
```

Select the desired characteristics, and sum together their corresponding numbers. This is the numeric value you use with **chmod**. For example, the correct value for a file that is **user** *readable* and *executable*, but only **group** and **world** *readable* is:

```
                  user    r      400
                          x      100
                  group   r       40
                + world   r        4
-------------------------------------
                                 544
```

You would change the mode on a file to this with:

chmod `544 file`

A.5 Specifying the Protection Mode with Symbols

A symbolic argument tells **chmod** to change relative to the current state. To use this you must specify the following:

- ***class***-- the group of people that the change affects

  u-- user

  g-- **group**

  o-- other(**world**)

  a-- all classes

- ***action***-- what type of change

  +-- add permission

  - remove permission

  =-- assign permission absolutely
  (resets other bits to no permission)

- ***permission***-- what type of permission

  r-- read permission

w-- write permission

x-- execute permission

X-- set to execute permission only if the file is a directory or has some form of execute permission set already. This prevents accidentally making a file executable.

The actual specification is a statement of the form:

chmod [*class*][*action*][*permission*] *file(s)*

Consider *myfile*, which has only **owner** *read* permission:

```
        -r--------  1 owner     92     Aug 23 22:22 myfile
```

First we will give everyone all permissions, and then remove **world** *write* and *execute* permission, and **group** *write* permission.

```
%ls -l myfile
        -r--------  1 owner     92     Aug 23 22:22 myfile
%chmod a+r myfile
%chmod a+w myfile
%chmod a+x myfile
%ls -l myfile
        -rwxrwxrwx  1 owner     92     Aug 23 22:22 myfile
%chmod o-w myfile
%chmod o-x myfile
%ls -l myfile
        -rwxrwxr--  1 owner     92     Aug 23 22:22 myfile
%chmod g-w myyfile
%ls -l myfile
        -rwxr-xr--  1 owner     92     Aug 23 22:22 myfile
```

This permits members of the same group to read and execute the file, and it permits the world to read it.

B. Course Software Lockers

If you're using Athena primarily for doing coursework, you'll need to access your course software, which is stored in a *course locker*. The programs that let you do this are *attach* and *detach*.

B.1 Attaching with *attach*

To get to your course software, you must attach a course software locker to your workstation. Once it's attached, the course software is a part of your workstation's file system. When you are finished using the software, you do not have to detach the locker from your workstation, although you can if you want to.

To attach a course software locker, use the *attach* command. Type **attach** ***lockername***. This command goes over the network to a remote filesystem server machine where the locker is stored and attaches the locker to your workstation at the */mit* directory. If a password is required, *attach* will prompt you for it.

```
athena% attach locker
locker: NFS mounted locker on /mit/locker (read-write)

athena%
```

Some course software lockers are attached in read-only mode. This means that you can run the programs in the locker, but you can't save any files in the directory where the locker is attached, nor can you modify any of the files in that directory.

B.2 Detaching with *detach*

After you use a course software locker, you can detach it from the workstation using the *detach* command. *detach* detaches the locker, and breaks the network connection between it and your workstation.

Before you use *detach*, you should check that you have used *cd* to move out of the directory that the locker is mounted in, in all of your windows. After you do so, you can use *detach* like this:

```
athena% cd
athena% detach lockername
lockername detached.
athena%
```

If you get an error message like:

```
athena% detach lockername
/mit/lockername: Device busy
Can't unmount lockername.
athena%
```

Check again that you are not in the mount directory, and that no processes are running from the locker. Then try *detach* again. If *detach* still fails, the workstation will force the locker off your workstation when it deactivates after you logout.

B.3 Attaching the Other File Systems

There are a number of lockers that contain some software that may not directly be related to coursework but is nevertheless useful, and in some cases can be very fun.

The *Student Information Processing Board* (*sipb*) maintains a file system that contains interesting software packages, such as *discuss* and *network news*. Note that these packages are supported by *sipb*, not by Project Athena.

You can *attach* the *sipb* file system with the command:

```
athena% attach sipb
```

This command attaches the file system to the */mit/sipb* directory.

Some other interesting lockers are **courseware**, **consult**, **postscript**, **beta**, and **games**. Some of their names are self-explanatory. Experiment with the less obvious ones to see what they contain. For a more comprehensive list of available lockers, attach the *info* locker and look at the file */mit/info/whats_available_where*.

C. More Fun with the X Window System

The X window system has a plethora of programs and assorted bells and whistles. You can fill your workstation's screen with all kinds of stuff--clocks, workstation load graphs, funky window icons. You can customize *xterm* windows and you can customize the behavior of the *uwm* window manager.

C.1 Customizing xterm Windows

There are many characteristics of an *xterm* window that you can change to make it look the way you want. Among other things, you can change: the size of the window; where the window appears on the screen; what font (typeface) appears in the window; and whether or not the window is displayed in reverse video (white characters on a black background).

There are two ways to customize the windows that you create with *xterm*. The first way is by giving *options* to the *xterm* command when you first type it in. The second way is by using *pull-down menus* to alter the characteristics of a window *after* you have created it. We will look at each of these ways in turn.

The first way to customize windows is by giving *options* to the *xterm* command. For example, suppose you want to create a window that uses the font called *vtsingle* instead of the default font (called *fixed*). To do this, just type:

```
athena% xterm -fn vtsingle &
[2] 8435
athena%
```

The hyphen in "**-fn**" tells *xterm* that what follows it is an *option*. The -fn option tells *xterm* to use another font instead of the default font.

Here are some other *xterm* options you might want to try:

-fn *font* Use ***font*** instead of the default font (*fixed*). You can list available fonts with the *xlsfonts* command and you can see what individual fonts look like with the *xfd* command. See the **Fonts** Section below. There are a lot of fonts available, although some of them are too fanciful to be useful.

-rv This option displays the window in reverse video, rather than in the default mode of black characters on a white background (like the login window).

-geometry *w*x*h*[[+-]*xoff*[+-]*yoff*]
This option lets you specify the dimensions of the window and (optionally) its position on the screen, instead of doing so with the mouse when you start up *xterm*. ***w*** is the width of the window in columns, and ***h*** is the height of the window in lines.

The optional ***xoff*** and ***yoff*** are the specifications of the upper left-hand corner of the window in "pixel coordinates". DEC workstations have screens that measure 1084 pixels across and 860 pixels down. The IBM RT workstations have screens which measure 1024 pixels across and 768 pixels down. On both workstation types, the screen's origin (0,0) is the display's upper left corner.

Pixel coordinates can be positive or negative. For example, to specify a window with 80 columns and 30 lines in the lower right portion of the screen, you could type **xterm =80x30+500+400 &**.

-t This option creates a window which emulates a *Tektronix 4010* terminal. Use the **-t** option when you are using *RS/1* and you want to see the graphics it produces.

To select more than one option, just add each one on the command line. For example, typing **xterm -fn vtsingle -rv &** will create a window with the *vtsingle* font and in reverse video. You can specify the options you want in any order. To read about the rest of the *xterm* options, look at the online manual page by typing **man xterm**.

The second way to alter window characteristics is by choosing them from a *pull-down menu*. To produce this menu, move the mouse cursor into the window whose characteristics you want to change, hold down the CONTROL key on the keyboard, and then hold down either the left or middle mouse button. The left mouse button menu is called **xterm X11** and the middle mouse button menu is called **Modes**. Drag the mouse cursor down the menu to the feature you wish to change, and then release the buttons. The feature will change immdiately. Experiment with the features to find out exactly what each one does.

C.2 The Fonts Section

As mentioned above, the X window system can display text in any of a large number of fonts. Use the **xlsfonts** command to get the current list.

The **xfd** command lets you display an individual font. The command works as follows:

`athena%` **xfd** *fontname*

where *fontname* specifies the name of the font.

xfd starts up in the same manner as *xterm*. Any of the *uwm* commands will work with an *xfd* window. To get rid of an *xfd* window, move the mouse cursor inside it and type **q** (for **q**uit). Because there are a multitude of font types, you probably want to list them in one window with the *xlsfonts* command, and use another window to issue the *xfd* commands.

C.3 Cutting and Pasting Between xterm and emacs Windows

The X window system lets you cut and paste text between windows. If you are running *emacs* in one window, you can get some text from another window and insert it into the *emacs* window. The *xterm* program provides you with a global cut buffer where you can store text that you want to move from one window to another.

Use the following procedure to cut text from an *xterm* window and paste it into an *emacs* window:

1. Move the mouse cursor into the *xterm* window and place it at the beginning of the text region you want to cut. Click the mouse's left button.
2. Place the mouse cursor to the bottom of the text region and click the right button. If you hold the button down long enough, *xterm* will display the region you are cutting in reverse video.
3. Move the mouse cursor into the *emacs* window. Move the input cursor to place within the *emacs* buffer where you want to paste the text. Press the middle button of the mouse. Voila.

There are other commands that you can issue to cut and paste text between one *xterm* window and another. See the *xterm* manual page for more information (type **man xterm**).

C.4 Customizing the Window Manager

There are many things you can change about your window manager. To do so, you need to create a *.uwmrc* file in your home directory. Type **man uwm** to find out about the format of the file and the things you can change.

C.4.1 Getting Rid of the Window Manager

You may want to get rid of the window manager because you want to try a different window manager, or try the same window manager with different options. Since *uwm* doesn't produce a window, there's nowhere for you to type **exit**, so you have to kill the *uwm* process with the Unix *kill* command. You can find out what that process's job number is with the Unix *jobs* command. Typing **jobs** in a window lists all of the background jobs (processes) that were started *in that window*. Since background jobs are usually started from your login window, move the mouse cursor into your login window and type:

```
athena% jobs -l
```

Unix will return something like:

```
[1]  +  8302 Running             uwm
[2]  -  8389 Running             xterm
[3]     8435 Running             xterm -fn vtsingle
```

Now, look for the *uwm* process and its corresponding job number. You see that *uwm* is your first background process. To get rid of it, type:

```
athena% kill %1
[1]    Terminated          uwm
athena%
```

Now you can start up another window manager with different options.

C.5 Additional X Programs

There are many other X programs besides *xterm* and *uwm*, some of which can do fun and quite useful things. You can display a clock with *xclock*, display a bar graph of your workstation's load average with *xload*, and change your workstation's display characteristics with *xset*. These programs let you set up your workstation's screen to suit your tastes.

Like *xterm*, other programs that create a window (like *xclock* and *xload*) have *options* that let you specify things like: where the window appears on the screen; how wide the window's border is; how far the border is from the text; what font to use for the text; whether to display the window in reverse video; and so on. The following program descriptions only list the most commonly used options of each. For a list of all the options, read the online manual pages by typing **man** ***program_name***.

C.5.1 Keeping Time with xclock

xclock creates a window that displays the time of day. You can use any of the *uwm* commands to manipulate this window. Since *xclock* runs continuously, you should run it as a background process:

```
athena% xclock &
[4] 8640
athena%
```

xclock starts up in the same manner as *xterm*. By default, *xclock* displays the time in analog mode (with hands).

Any of the *uwm* commands will work with an *xclock* window. *xclock* options let you specify the window's location on the screen, whether to display it in reverse video, the icon font, the border width, the update frequency, etc. For a complete list of options, look at the online manual page by typing **man xclock**.

C.5.2 Monitoring the Load Average with xload

xload displays the workstation load average as a bar graph over time. The graph is updated every five seconds. Each horizontal line on the graph represents one unit of load average. The graph will rescale and redraw itself when the bars scroll off the window. Since this program runs continuously, you should run it as a background process as well. So just type:

```
athena% xload &
[6] 8671
athena%
```

xload starts up in the same manner as *xterm*. You can use any of the *uwm* commands on an *xload* window.

xload options are roughly the same as the ones for *xclock*. See the online manual page by typing **man xload**.

C.5.3 Changing the Background Screen with xsetroot

Another X program you might want to try is *xsetroot*. This program lets you change the pattern of the background screen. Here are some of its options:

-gray or **-grey**	Makes the entire background gray (or grey). It's easier on the eyes.
-mod ***x y***	Creates a grid pattern on your screen. ***x*** and ***y*** are integers ranging from 1 to 16. Try different combinations of numbers. Don't use zero or else you'll get core dumped. Negative numbers are taken as their absolute value.
-bitmap ***file***	This option turns the background screen into a repeated sequence of *bitmaps*. A *bitmap* is a 16x16 array of black and white pixels. ***file*** is a *bitmap file*; it's created with the *bitmap* program. Bitmaps are easy to create, and they can look like whatever you want: happy faces, your initials, clever patterns, etc. Be creative. Have fun. Type **man bitmap**.
-cursor ***cursorfile maskfile***	This option lets you specify what the mouse cursor looks like when it's over the background screen (i.e., when it's not inside any window). ***cursorfile*** and ***maskfile*** are bitmap files (see **-bitmap** above). You should probably keep the ***maskfile*** bitmap all black until you get used to how this option works.
-rv	Exchanges black pixels for white pixels, and vice versa. It can go with any other option.
-def	Restores the background to the default grey mesh.

NOTE: Use only one of the **-solid, -gray, -grey, -bitmap**, or **-mod** options.

C.5.4 Changing Your Display Characteristics with xset

Your workstation has display characteristics such as keyclick volume, bell volume, and mouse speed. You can change them with the *xset* program if you do not like the default settings. There are two ways to do this: by giving *command options* to *xset* or by choosing characteristics from a *pull-down menu.*

The first way to change the display characteristics is by giving *command options* to *xset.* Here are some of the more popular ones:

c	Keyclick: turn off (c 0) or set the volume (c [1-100]); default is off.
r	Autorepeat: turn on or off; default is on.
b	Bell: turn off (b 0) or set the volume (b [1-100]); default value is 50. You can also set the bell's pitch and duration.

To set the options, type the ones you want on the command line, and follow each option with the appropriate value. For example, typing **xset r off c 50 b 75** turns off autorepeat, changes the keyclick to 50, and changes the bell volume to 75. There are a few options that are obscure. For more information, check the online manual page by typing **man xset**.

Note: **xset** no longer controls the shift LOCK key. To alter its behavior, use **xmodmap**.

You can also display characteristics by using a *pull-down menu*. To produce this menu, hold down the SHIFT key, the META key, and the middle mouse button. (The mouse cursor can be anywhere on the screen.) The **WindowOps** menu will appear, but by moving the mouse cursor to the left or right, it will change into another menu called **Preferences**. Drag the mouse cursor down this menu to select the display characteristic you want, and release the buttons. The characteristic will change immediately.

C.6 Printing Out an X Window

You can get a printout (or *dump*) of an X window; this lets you get hardcopy of any graphics that you produce on your workstation.

There are two ways to get a window dump. There is a fast method that uses the *xdpr* command or a slower, more controlled method that uses the *xwd*, *xpr*, and *lpr* commands.

This procedure differs depending upon whether you want to format your window dump for a Postscript printer (LN03R or LPS40) or a non-Postscript printer (LN03+ or 3812 Pageprinter). By default, the window dump will be formatted for a non-Postscript printer. To format it for a Postscript printer, you must specify the appropriate options. This is true regardless of which method you use. Be aware, though, that window dumps will print out *much* more quickly on a non-Postscript printer.

Consult the document **Printing from an Athena Workstation** for more information about printing and the X window system.

C.6.1 Fast Method

The fast method is to use the *xdpr* command. Simply type **xdpr** in a free xterm window, and the mouse cursor will change into a crosshair. Move the mouse cursor into the window you want to dump and click any mouse button. *xdpr* will beep once when it begins to copy the window and twice when it is done. For example, to produce a window dump on the LN03+ printer located in Building 11, use

`xdpr -P`*europa*

Wait a while, and then pick up your printout at the Dispatch Desk on the second floor of Building 11. By default, *xdpr* tries to make the picture as big as possible, and orients it in a way that depends upon the dimensions of the window.

Use the *-Pprintername* option to specify a different printer. Use *-device ps* option to format the window dump for a Postscript printer. Use *-device pp3812* option to format the window dump for a 3812 PagePrinter.

C.6.2 Slow, Controlled Method

Use this method if you want to dump a window to a file. This is useful if the window you've created is important to you, or if you want to experiment with its appearance on paper. This method lets you specify the scale, size, and orientation of the window as it appears on paper.

There are three steps to this method, and one command for each:

1. **xwd -out** ***filename1***
   xwd is the X window image dumper. It stores a window image in a specially formatted window dump file called ***filename1***. It changes the mouse cursor, etc., the same way *xdpr* does. You can *undump* the file back to the screen by typing **xwud -in** ***filename1***.
2. **xpr -output** ***filename2 filename1***
   The *xpr* command formats ***filename1*** for output on an LN03, and puts it in ***filename2***. It lets you specify things like the window's scale, size, and orientation. Use the *-device ps* option to format ***filename1*** for a Postscript printer. Use the *-device 3812* option to format ***filename1*** for the 3812 PagePrinter. For more details, type **man xpr**.
3. **lpr** ***filename2*** The *lpr* command sends ***filename2*** to the LN03 in your cluster, unless you specify otherwise with the *-P* option.

You can get a dump of the entire screen by choosing the background screen as the window to dump.

C.7 Customizing Your Environment with Dot Files

Your *dot files* are the files in your *home directory* that begin with a period or "dot" (.), such as *.login*, and *.cshrc*. All new Athena users are provided with a set of default dot files in their lockers. To get a list of these files, get into your home directory and type **ls -a** (for **lis**t **a**ll).

Each of these dot files sets certain aspects of your workstation's environment, and you can customize your environment by modifying them. If you want to do so, you'll have to do your homework first to learn about the things you are modifying. You should avoid deleting most of the things in these files, as doing so may jeopardize the integrity of your NFS filesystem.

Each of your dot files does many things, but in general, your *.login* file sets your *environment variables*, and your *.cshrc* file sets your *shell variables*. For information about changing these files, read the online manual pages on *csh* and see the Athena document ***More Unix: The C Shell***.

C.8 Where To Go From Here

This chapter and the manual pages for the individual X programs are the only sources of information about using X. A nice overview of X is in the X manual page, which you can get by typing **man X**.

There are other X programs that are not discussed in this chapter. They include *xrefresh*, *bitmap*, *ico*, *xcalc*, *muncher*, and a few others that are less frequently used. You can read about them in the online manual pages by typing **man *program_name***.

There are also some aspects of X that are beyond the scope of this chapter. For example, you can do some *shell programming* to set up your screen automatically when you login; read the online manual page on *csh* or the document ***More Unix: The C Shell***. If you're the daring type, you can write programs in the C programming language that use X and the mouse. For example, you could write your *own* window manager instead of using *uwm*. In order to do this, you need to include the X Library in your program. To find out more, look at the document ***Xlib - C Language X Interface: Protocol Version 11*** by Jim Gettys, Ron Newman, and Robert W. Schiefler.

Table of Contents