APS 105F – Computer Fundamentals

Assignment #1: Basic UNIX

Fall 1997

(To be completed in your lab period Sept. 10/11)

1 Introduction

This first lab session will serve as your introduction to the lab workstations and the UNIX operating system. The assignment is in the form of a tutorial and will gradually take you through the basics of interacting with the computer so that you may quickly master them. You may find that this assignment will take longer than two hours to complete. If this is the case, please complete it at another time, but as soon as possible.

The *Getting Started With UNIX* manual is found online at

http://www.ecf.toronto.edu/ecf/docs/.

You are encouraged to continue your introduction to the laboratory computing environment by reading the manual after you complete the assignment. This tutorial includes references to the manual for further details. For even greater in-depth understanding on UNIX, there are many fine introductory texts available.

2 Logging Into The Computer System

The first thing you must do when you begin your work on the computer is to "log in" to let the system know that it is you who is using the system. Not just anyone can use the computer network. Access to the system is controlled to maintain its security; only those with accounts on the engineering computing facility (ecf) network can access it. All undergraduate electrical and computer engineering students are automatically given an account when they register, so you should be able to login.

To login, you must have two pieces of information: a login name and a password. Your login name (or "login") is a 6-8 character string that is your identity on the computer network. Your password is a variable length character string known only to you to maintain the security of your account[1]. It serves as a key to your account much like you would use

[1]In fact not even the system knows what your password is, so don't ever forget it.

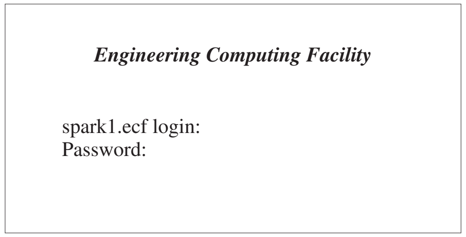

Figure 1: The window for logging in

a key to gain entry to your locked house. Initially your password is set to your student number. You will change your password to something less obvious later in this assignment, and keep it as a secret only to you.

When you sit down at one of the X Terminals, you will see a greeting window that looks like the one shown in Figure 1. Type your login name after the "login:" prompt, press the RETURN key, then type in your password[2] and press RETURN again. Your lab instructor will help you to find out what your login name is. If the message "Login Incorrect" is displayed, try retyping your login and password. If you have typed everything correctly, the screen will clear, and another window will open up. If not, consult a teaching assistant.

Just as you login to the computer when you begin to use it, you must logout of the computer when you are finished using it. If you must logout before you reach the end of the assignment, read the final section of the assignment which instructs you on how to logout.

3 The X Window Environment

After logging in, the X Window computing environment is automatically started. You will see as you progress in this course that the X Window environment is a powerful one to work in because of the ability to have the computer do different things at the same time, a concept known as multitasking. The X Window environment also allows you to conveniently organize the windows on the screen. For now, we will learn the basics about manipulating windows. Some additional information can be found in Section 4.3 of the manual.

The manipulation of windows on the screen is accomplished mainly through the use of the mouse. If you are not familiar with how to use a mouse, you may want to ask your neighbor or the teaching assistant (T.A.) for a quick demonstration and some help with the terminology that follows.

When X Windows starts up, there will be a single window on the screen that appears like the one in Figure 2. (There may also be a smaller window titled "Message of the Day". After you read this, close it by clicking the "Close" button.) From what is written in the title bar, the thin rectangular box at the top of the window, we can see that this is the Login Session window. A window can be moved around on the screen by clicking on the title

[2]For security, the characters in your password will not appear as you type them.

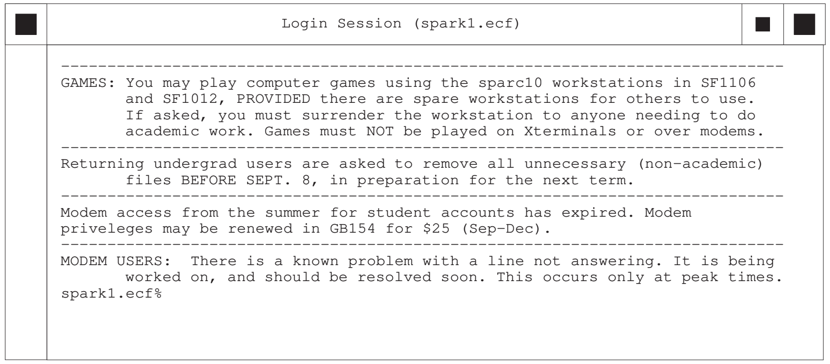

Figure 2: First Xterm window: the Login Session window

bar and dragging it to another location. Give this a try. You can also change the size of a window using the mouse. Move the cursor to a corner of the window. You should see the cursor change to an L-shape that aligns with the corner of the window. Click the mouse and drag the cursor away from the window, then release the mouse button. This action increases the size of the window. The window could similarly be decreased in size by dragging the edge of the window inward. At the top right of the window are two little boxes that are also used to control the size of the window. Clicking the box on the right maximizes the window so that it occupies the entire screen. Clicking this box again returns the window to its original size. The other little box is used to reduce the window to an icon. Click on this box to iconify the window. The icon can now be moved to another location on the screen by clicking and dragging it. Reducing windows to icons is an important way to organize the screen when you are working with many windows. You can restore an iconified window to its original size and location by double clicking on the icon.

Now look inside the Login Session window.[3] You should see some introductory messages about the system followed by the UNIX *prompt*:

`spark1.ecf%`

The prompt will be some variation of this depending on the name of your computer. You can issue commands directly to the UNIX operating system at this prompt. In this assignment, the commands that you are to enter are shown in boldface. The first command you will enter will tell UNIX to create a new window. Move the pointer with the mouse so that it is on top of the Login Session window and enter the following command:

`spark1.ecf%` **xterm &**

[3]The computer may be asking you if you want to read some messages. It is expecting you to respond with y, n, or q. Enter q to quit.

The xterm command stands for X Terminal and it creates a window which, like the Login Session window, has a UNIX prompt at which you can enter commands. You can now use either the Login Session window or the new *xterm* window to issue UNIX commands. You can create additional windows with the "**xterm &**" command if you like and have several windows on the screen. Despite having many windows on the screen, only one window is active at a time. The active window has a highlighted border, and all text typed at the keyboard is directed to the active window. You can make another window active simply by pointing inside it. Sometimes, windows will be partially or completely hidden behind other windows. To bring a window to the top, point at its title bar and click the left mouse button. To push a window to the bottom, point to the title bar and click the right mouse button. Practice these techniques by overlapping your windows and raising them and lowering them in turn.

A window is closed by double clicking on the small box at the top left corner of the window. Try this on an Xterm window. Do not try this on the Login Session window, because if you do, you will be logged out of the computer. For this reason and others, it is good practice to iconify the Login Session window and to work in other windows, so that the Login Session window is kept intact.

When you create new windows with the **xterm** command, the ampersand is an optional component of the command. By default, commands are executed in the *foreground*. Adding an ampersand to the end of a command instructs UNIX to execute it in the *background*. What does all this mean? Commands that are executed in the foreground must complete execution before the terminal that called it can do anything else. So if you call up a new window with the **xterm** command and do not include an ampersand, you will not be able to use the window from which you called up the new one until the new terminal is closed. When a command is executed in the background, the terminal from which it was called can continue to accept commands while the background command is executing. The bottom line is that when calling up new windows from the command line, add an ampersand at the end of the command so that you can use both the new and the original windows. Try calling up new windows using the **xterm** command with an ampersand and without an ampersand and observe what happens in the original window.

4 Changing Your Password

At this point an important task that you must do is to change your password. Remember that the password to your account is the only security measure keeping your account private so you should choose your password carefully and keep it an absolute secret. You change the password by using the **passwd** command. Your password should be at least 6 characters long, with 7 or 8 being preferable, and it should contain at least one non-numeric character. Do not use the special characters DELETE or BACKSPACE in your password. Try and use a mixture of upper and lower case letters, which makes it less likely that someone will guess your password. In particular, do not use any single words that could be found in a dictionary or names.

When changing your password, the dialog should look like this:

`spark1.ecf%` **passwd**

Changing password for user: <your login name>
Old password: <**type in your current (old) password here**>
New password: <**type in your new password here**>
Retype new password: <**type in your new password again**>

As you type both the old and new passwords, the characters will not be visible on the screen. This is deliberate, to prevent others from seeing your password by looking over your shoulder. Because you cannot see your password, you are asked to type it twice, on the theory that you are unlikely to make the same typing mistake twice.

5 Organization Of Files

You may have heard at some point that computer systems consist of *hardware* and *software*. Hardware refers to all the physical components such as microchips, wires, monitors, etc., while software refers to the programs that are stored, accessed and executed using the hardware. For the lab assignments of this course you will be concerned only about software.

Software is stored as data and data is organized into what are called *files*. There are many types of files. Some files are programs that can be executed, other files contain text that can be read, while other files contain images that can be sent to the screen. Two things that all files have in common are that they can be stored in computer storage, and that they all have a name. We will soon take a look at some files in the computer system. As we add more files to computer storage the number of files grows to the point where it is hard to keep track of them all. For this purpose, files are organized into *directories*. You can think of directories as being like file folders. File folders are used to organize papers and articles and things and are named according to their contents. Computer directories are similar in that they organize files by grouping them. However, computer directories are even more versatile in that they could contain other directories as well as files.

If you mapped out the directory and file organization onto a graphical representation, you would get a tree-like structure as shown in Figure 3. At the top is the root directory. It is represented by a slash character, which is its name. The root directory may contain files, and it will also contain other directories. The files and directories within the root directory are shown as nodes below and attached to the root directory. These directories may have files and other directories contained within them and these are shown as nodes below and attached to the *parent directory*. This pattern can be repeated many times.

Now remember, it is the files that are actually important when it comes to running programs. The directories are only there to organize the files. When you want to work with files, you generally have immediate access to the files in one directory only. This directory is called the *working directory*. To find out what the *present working directory* is, type in the following command:

`spark1.ecf%` **pwd**

UNIX will respond with a string that looks like `/u/0T1/yourLogin`. This string describes a path through the directory tree from the root directory to the present working directory.

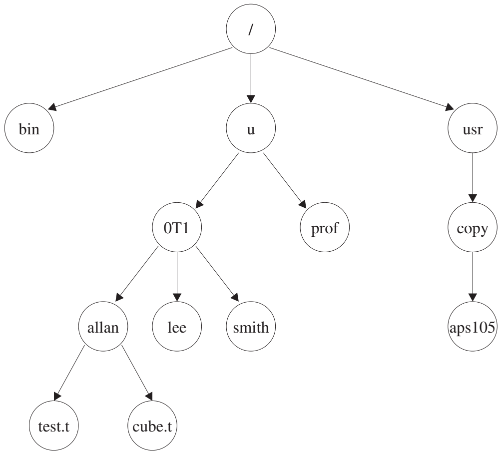

Figure 3: Unix file organization

Reading it from left to right, the path starts at the root directory, which contains the directory named `u`, which contains the directory `0T1`, which contains the directory with the same name as your login. What this string is saying is that there is a directory with the same name as your login name and it is located in the path described above and it is the present working directory.

You can change the present working directory using the **cd** command. The **cd** command takes an argument, which is an extra term that is input after the command. In this case the argument is the path of the directory that you want to change to. There has to be a space between the command and the argument. Try the following:

`spark1.ecf%` **cd /**

Then enter the **pwd** command. You will see that the present working directory is as you have requested - the root directory. Now try the following:

`spark1.ecf%` **cd /u**

Entering the **pwd** command will show that you are now in the directory requested. Finally, enter:

`spark1.ecf%` **cd /u/0T1/yourLogin**

and this will take you back to your home directory as you can verify using the **pwd** command once again.

Up until now, we have been specifying the path from the root directory to the desired directory as the argument to the **cd** command. This is called the absolute path. It is not necessary to always give the absolute path as the argument and usually, one does not do so. Rather, one can use what is known as a relative path. Try this, go to the root directory by entering **cd /**. Now go to the `u` directory by entering the following command:

`spark1.ecf%` **cd u**

If you do a **pwd** command, you will see that you are now in `/u` even though you didn't have the slash beginning the argument of the **cd** command. Because the argument did not begin with a slash, UNIX understood you to mean go to the `u` directory from the present working directory, which is /. Now try to go to the `0T1` directory by entering:

`spark1.ecf%` **cd 0T1**

Again, if you enter the **pwd** command you will see how you have moved into the new directory.

There are two important short-forms that are used to specify directories. Two consecutive periods (..) refers to the parent directory of the current directory, so if you use the command **cd ..** the parent directory will become the working directory. The other important short-form is the ~ character, which refers to your home directory. No matter where you are in the directory tree, entering **cd ~** will make your home directory the working directory. With the cd command, you can get even simpler than that and enter just **cd** to make your home directory the working directory. Practice using these short-forms. First, enter a command to go directly to your home directory **(cd ~ or cd)**, then back up the directory tree, one directory at a time, using **cd ..** Then return to your home directory once again.

Now you should have a good understanding of navigating around directories. But of course, the interesting part is not directories, but what they contain. To determine what a directory contains, one uses the **ls** command, which stands for "list" and instructs UNIX to list the contents of the directory, which will include any files and subdirectories. Now let's see what we can find using the **ls** command. Go to the root directory and enter:

`spark1.ecf%` **ls**

You will see some strings arranged in columns. The **ls** command is displaying the names of files and subdirectories contained in the root directory. Try using **ls** in the `/u` directory, and in other directories if you wish. Now go to your home directory and enter **ls**. There is probably no output. This is because as a new user, you have not created any new files or directories. However, even though **ls** does not show you any files in your home directory, there are some there that are *hidden*. To show all the files, including those that are hidden, enter:

`spark1.ecf%` **ls -a**

Some filenames will be displayed. The names of hidden files are easily distinguishable as they all begin with a period. When your account was created, your home directory and a few hidden files were automatically created for you. These files are hidden because you would not normally work with them as you would with your program files. Hidden files usually

contain special information for the computer system on various things like how to set up your screen display, or how to respond to your commands. You may want to explore some of these files and determine what they do on your own time.

One last variation on the **ls** command that you will probably find helpful is:

`spark1.ecf%` **ls -al**

The final **l** stands for long form and entering this command will show all files and directories with more descriptive information for each item. A concise summary of directory and file organization can be found in Section 5.2 of the manual.

6 Manipulation Of Directories and Files

In the last section, it was shown how to explore the directory tree and how to look at existing directory contents. Now you will learn the basic commands for changing directory and file organization. For the most part, this will be constrained to directories and files within your own home directory.

Changing the directory tree structure is accomplished through the use of two basic commands: one that creates new directories, and one that destroys directories. The mkdir command makes a new directory. To illustrate its use, go to your home directory, then enter

`spark1.ecf%` **mkdir aps105**

The argument supplied with this command is the name of the new directory, "aps105". You can verify that a new directory was made by using the **ls** command. Notice that by default, the new directory is placed within the present working directory.

Removing directories that are no longer required is just as easy. To do so, you go to the directory in which the unwanted directory is contained, then use the command:

`spark1.ecf%` **rmdir directoryName**

A directory cannot be removed unless it contains no files and no other directories.

Basic file manipulation is achieved through the commands that copy, move and delete files. The **cp** (copy) command is used to make a duplicate of an existing file. It takes two arguments: the first argument is the name of the file to be copied, the second argument is the new name of the new duplicate file. In your home directory, enter the following command:

`spark1.ecf%` **cp .login new.file**

This command has made a copy of the hidden file ".login" and called it "new.file" which is not a hidden file because its name does not begin with a period. Verify that the copy was made using the **ls** command.

The **mv** (move) command is similar to the **cp** command in that it copies a file and gives it a new name, but while the **cp** command preserves the original file, the **mv** command deletes it. **mv** can be used to move a file to a different directory, and this will be demonstrated later, but **mv** can also be used just to rename a file. Try the following:

spark1.ecf% **mv new.file newname.file**

Again, the **ls** command will show you the change that was made.

Finally, files can be deleted by using the **rm** (remove) command. Delete the new file that you created with the following:

spark1.ecf% **rm newname.file**

Up until now, the file and directory manipulation commands have been demonstrated to operate only within the present working directory. These commands are in fact more versatile than this because it is possible to work with files and directories other than those in the present working directory. To work with a file or directory outside of the present working directory, you must supply its path rather than just its name for the command argument. Absolute or relative paths can be used.

For example, **cp** can be used to make a duplicate of a file that is not within the present working directory. Use the following command to copy the Turing program demo.t which is located in the directory `/usr/copy/aps105` to your home directory:

spark1.ecf% **cp /usr/copy/aps105/demo.t .**

The first argument of the command is the absolute path of the demo.t program file. The second argument is a period. This is a short form commonly used in UNIX that represents the present working directory. UNIX interpreted this command to mean place a copy of the file demo.t (located in `/usr/copy/aps105`) in the current directory and give it the same name (demo.t). Now try using the **mv** command to place this file in the aps105 directory you created earlier, and to rename it "test.t":

spark1.ecf% **mv demo.t aps105/test.t**

The **mkdir, rmdir,** and **rm** commands also can take paths as arguments. Experiment with these commands some more until you get the hang of it. Use Section 5.3 of the manual as a reference.

7 Text Editing

The most basic and most important files that you will be working with are *text files*. A text file consists of readable text that can be created using the keys on a standard keyboard. Text files are used for various purposes so you will want to have a good knowledge of how to work with them. In this course, you will be making text files that will serve as Turing programs.

You can create or make changes to a text file using what is called a *text editor*. There are several text editing programs available for use on the computer network. The simplest editor is called *xedit*. Call up xedit on your computer using the command:

spark1.ecf% **xedit &**

The xedit window that pops up is a different kind of window from the xterm windows we have been using. This window is specifically designed for editing text files. There are four sections to this window. The bottom section is a large empty box and it is here where the text file is composed. Xedit starts up with this section cleared so you can create a file from scratch.

Let's create a simple text file. Move the mouse so that the cursor is pointing in this section. Then write some simple text in this box such as "Creating a text file is so much fun.".

At the top of the window is the filename section. There are three boxes at the left with the words "Quit", "Save" and "Load". Beside this is an empty box. This is the space where the name of the file is specified. Because this is a new file, it has yet to be named. Point the cursor inside this space and type "fun.txt", then click on the Save button. You have just created a new text file. Now click on the Quit button to close the xedit window.

To see that a new text file was created, go to the xterm window and enter an **ls** command. The file `fun.txt` should appear in the directory listing.

Open up an xedit window again using the command written above. Besides creating new files, xedit can be used to change existing files. In the filename space, type "fun.txt", then click on the Load button. The file you created is loaded into the text file section. You can now edit the file. Point with the mouse just after the word "Creating" and click. Now press the Back-space button on the keyboard several times until the word is erased. Now, type in the word "Editing". By clicking on the Save button, you can save the changes you have made. Finally, click the Quit button to exit the editor.

Editors are useful for creating and editing text files, but if you just want to read the contents of a text file, you can use one of several UNIX commands at an xterm window. This saves you from having to start up a text editor to view the file. The most convenient command for typing out a text file is the **more** command. Entering **more** along with the text file name as an argument types out the file in the xterm window, one screenful at a time, if the file is longer than one screen. You can advance through the file by pressing the space bar. Try using the **more** command as follows:

spark1.ecf% **more fun.txt**

For information on other commands that handle the contents of files, see Section 5.4 of the manual.

8 Electronic Mail

One of the advantages of computer networks is that it provides a means of communication between users. Electronic mail, or "e-mail", is the most important method of communication, in which messages can be composed and sent to specific recipients, just as with regular mail. Experiment with e-mail by sending a letter to a friend or to someone sitting close by in the lab. Find out what his/her login is. Now, you can send them an e-mail letter by following these instructions:

1. Enter the command:

spark1.ecf% **mail < yourFriend'sLogin>**

2. The mail command will prompt you for the subject of the letter. Type in a short description of the subject and then hit RETURN.

3. Type in the letter.

4. After the closing salutation hit RETURN. Type in a single period and hit RETURN again and the letter will be sent.

If at any time while you are composing the letter you decide to cancel it, enter ^C twice; that is, push the CONTROL-key (as you would the shift key) while typing in a C in lower case. (The ^ character is a symbol for the Ctrl key.)

If someone has sent mail to you, UNIX will notify you with the message `You have new mail`. To read new mail, enter the command:

spark1.ecf% **mail**

The UNIX mail facility will display to you a list of the letters waiting to be read. For each letter in the list, the sender's login and the letter's subject will be displayed. To read the letters all you need to use is the space bar. Press the space bar to display the first letter in the list and continue to use the space bar to advance to other letters when you are ready. This is the most basic way to use e-mail, but more control over the program is possible. For more information, read Section 5.13 in the manual.

You will soon realize that e-mail is the most reliable method of getting in touch with others on campus, and indeed, with users of computer networks all over the world. Because ecf has access to INTERNET, sending and receiving messages is as easy with users on the other side of the planet as it is with your friend on the other side of the room. All you need to know in addition to the user's login is the INTERNET *address* where he/she can be reached. When you send mail, you append the address to the login as follows:

spark1.ecf% **mail <login@address>**

Mail is used of course for more than just sending friendly letters. It is the easiest way to get in touch with professors, technicians, T.A.s and others if you have questions or requests to make. Make sure that you get the e-mail addresses of the professors and teaching assistants of your courses. If you have suggestions about APS105, mail a message to `pc@eecg`.

9 Continuing Your Introduction

This tutorial has covered a lot of ground but it is only the tip of the iceberg. You are encouraged to continue learning about the UNIX operating system and the computer network. First of all, learn in greater depth about the topics that were covered in this assignment from the sections of the "Getting Started With UNIX" manual that were referenced in the tutorial. Practice and experiment with the various commands and techniques. You will eventually

want to learn most of the information presented in the manual. Sections 1 through 4, and sections 5.1–5.5; 5.8–5.15 are of immediate interest to all users. For those of you who are hacker wanna-be's, you may want to explore some of the program packages on the system in Section 5.7; the facilities used on the INTERNET such as *News, ftp, telnet* and the World Wide Web (WWW) in sections 5.16–5.18; or the *vi* editor that can be accessed through the ECF documentation page.

If you want in-depth information on any particular UNIX command or program used on the system, use the **man** (manual) command, with the command or program name as the argument. For example, enter

spark1.ecf% **man ls**

to get information on the **ls** command. As the information is displayed, you will see –More– written at the bottom of the screen. At this point, you can press the space bar to display the next screen, or press **q** to quit.

10 Browsing the WWW

If you are not familiar with the WWW, you can find an introduction at

http://www-medlib.med.utah.edu/travel/

Start the netscape browser:

spark1.ecf% **netscape &**

Click on the **Open** button and type in the URL above. Then click on the **Open in Browser** button.

Enjoy! But don't forget to work on your assignments too!

11 Logging Out Of The System

When you have finished your session on the computer, you must logout by entering the following command in the **Login Session** window:

spark1.ecf% **logout**

If you forget to do this, someone will be able to get unauthorized access to your programs and other files after you leave the terminal. They could get you in **BIG** trouble.