A Crash Course on CSC[1] Facilities[2]

by Peter Alfeld, 1-6842, JWB 127, alfeld@math.utah.edu

Purpose I assume you know nothing about computing, except that nowadays it runs on electricity. The purpose of this handout is to get you as soon as possible to a stage where you can use our facilities effectively and productively. I assume that your use is typical, i.e., you wish to

— use electronic mail,

— plot the graphs of functions

— manipulate symbols, e.g., differentiate and integrate functions, or factor polynomials

— edit files,

— typeset documents, and

— write and run small programs,

in that order of priority. Topics are arranged roughly in the sequence in which you might typically need them. So the farther you read the more you will be able to accomplish. At first you might think that as you go along things only get more *esoteric* although in just a few weeks you will look at the material in these notes as gratifyingly elementary. Also note that the information given gets sparser as you move along. This is because you are getting more sophisticated and independent, not because I'm getting lazy or running short on time. Not counting the amount of time it takes to learn TEX or programming, but including the time needed to get going with Emacs, you should be able to work through this material in about three fun filled days which you may fondly remember for several decades. I still remember vividly their equivalent for me twenty years ago.

Suggestions I'm planning to keep updating and improving these notes. So let me know what you think about them, and what should be added, deleted, or modified. Leave a note in my box (Peter Alfeld), call me (1-6842), or (once you know how) send electronic mail (to *alfeld*, or, if you work from a non-CSC-account, *alfeld@math.utah.edu*). Electronic mail is the preferred way!

Revisions To obtain the newest version of this handout print the postscript file

/u/ma/alfeld/public/crashc.ps.

Disclaimer In computing, like everywhere else, there are many ways of accomplishing a given task. To save time these notes will show you just one way for each task. There may be better ways and you may have a friend (not my friend) who tells you that everything here is wrong, complicated, awkward, or dangerously oversimplified. That just means he got started differently, or forgot what it is like to get started. But as a matter of habit you should keep on the lookout for shortcuts and new ways of doing things. There are all kinds of nifty gadgets, tools, utilities, and major software packages that keep computing fun and that may actually save you time and help you accomplish your purposes.

Keep Talking As you need to know more about computing don't hesitate to ask anybody who is close by. Many times there will be something you want to do which somebody can tell or show you in a minute, but which would take you days, weeks, or forever to figure out by yourself.

[1] i.e., Center for Scientific Computing

[2] TEX processed September 26, 1994

No matter how intimidating they may look, people are usually eager to help, and before too long you will be able to help others and pass on the tradition.

A Crossroads. Read This! You can use our systems in one of two modes: working with a workstation as a *console*, or connecting to a workstation over *the network*. In the first case you have a sophisticated window system at your disposal, but you must go to a specific physical location (JWB 120, Building 129, or, if you are lucky, to your very own workstation in your office). To learn about working in this mode, simply proceed to the next item and read on. In the second case you have only one or a few windows (with limited capacities) open, typically on a Macintosh, or another workstation. However, you can access our systems via the telephone[3] or from any place that is connected to the network. To work in this mode you don't need to read about our window system, so move to the headline called *Getting down to Work* (a few pages down).

What to do We assume you have been given a user name and an initial password. To get the most benefit from these notes locate a workstation (in JWB 120 or Building 129), sit down in front of it, and play with it as you read on. It's important to try out things as you go along. Feel free to explore. You won't be able to damage anything merely by typing commands, so be unafraid! When I watched the deck of cards carrying my first program move into the card reader I fully expected bells to ring and smoke to rise. This just doesn't happen.

The Hardware In front of you are three pieces of hardware:

— The *Keyboard,* i.e., the object with many buttons, including a number of strange ones,

— The *Mouse,* i.e., the funny little contraption with three buttons.

— The *Monitor,* i.e., the big box obscuring your view and including a large piece of glass (which is the *screen*).

Together with the box underneath the monitor these items constitute *your workstation.*

Computing On the simplest level, computing is all about manipulating the first two items to get the third to show what you want to see. If you are in Building 129 that will be in black and white, otherwise it may be in color. You are the *user.* The workstation is there to serve you.

The infrastructure Your workstation is only the tip of an ice-berg. Underlying it is supporting hardware and an electronic network that allows everybody to interact with everybody else. For example, your data are kept on a magnetic disk that you access through two other machines. The advantage of this system is that you can work with your data from any machine without having to transport data physically. It is also more efficient since information needs to be stored only once, and it makes it easier to upgrade and maintain our software. The departmental network is illustrated in Figure 1. It in turn is only a tiny part of a network that spans the entire globe. If you have the appropriate login privileges you can simultaneously use machines in Sydney, Provo, and Moscow!

Logging In We assume that your machine is turned on and that it is ready to start interacting with a user, even a brand new one like yourself! Whatever else you are going to do, **don't**

---

[3] Dial 581–8105, wait for the LOGIN prompt, login with your usual name and password, and then type

```
rlogin xxx
```

where `xxx` is the name of the workstation you want to use (e.g., sandy, magna, solitude, cayley, or your private workstation). Our modems work at 300, 1200, 2400, 9600 and 14400 baud.

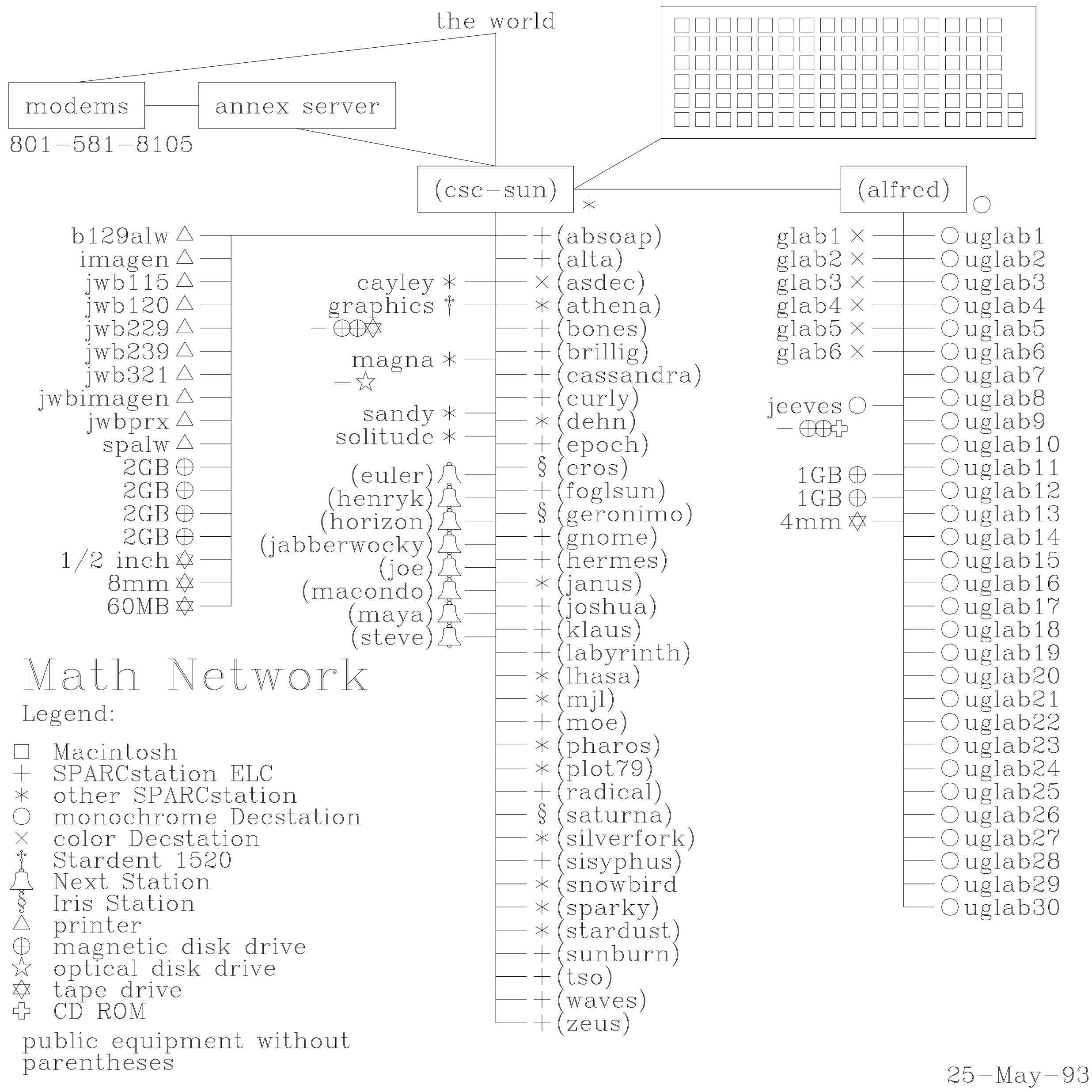

Figure 1. The CSC Network.

turn off the power[4]**!** (I won't even tell you how to do that, although you might possibly be able to figure this out by yourself.) If you get stuck, ask the Lab Assistant, or call 1-5257 and report your trouble. If that doesn't work leave a note on the keyboard explaining what happened and requesting that somebody *reboot* the machine. Then go to

[4] If you do turn off the power you interrupt all jobs that somebody else may be running on the machine. You may also damage the file structure which may bring down other systems, thermal contraction and subsequent expansion of the power supply may shorten the life span of the workstation, and if the station fails to boot upon being turned back on you have rendered it unusable to all users until an expert attends to it. **You also automatically lose all user's privileges.**

another workstation and try again. (If you mess that one up too, give up for the time being and at least until you have talked to someone more knowledgeable who can tell you what went wrong.) On the screen you should see a display much like this[5]:

Welcome to the X Window System

Login:

Password:

Your first task is to *log in*, i.e., to introduce yourself to the machine and persuade it that you are a legitimate user. That's called *logging in.* You do this by specifying your user name and authenticating it by a *password* that you set and that should be known to you alone.

RETURN key Find the button (or *key*, as we will call it from now on) labeled RETURN on your keyboard. *Hitting* it (i.e., depressing it, and letting go) usually means you want the machine to do something. Often, when you don't know what else to do hitting RETURN will get things going again. Don't hit RETURN yet, though!

Delete key An almost equally important key is in the upper right corner of the main part of your key-board. It's marked with a box that has an X in it. Pressing that key causes the last keystroke you made to be annulled. You can use it to correct errors.

User Name There is a black vertical line called the *cursor* on the screen to the right of *Login:* or of *Password:*. In the latter case push the RETURN key. If there is any writing after *Login:* press the RETURN key twice. It should now be to the right of *Login:*. Carefully type your user name. If you make a mistake use the key labeled *Delete* to erase whatever you typed mistakenly. When you are done read over your user name and verify it's correct. Then hit RETURN once. The cursor moves to the right of *Password:*.

Your Password Now type your password. Note that it doesn't show up on the screen. This is so that people can't look over your shoulder and read it. Proper computer etiquette requires that you avert your eyes when somebody else is typing a password so you can't tell what she is typing by watching the keyboard. You can use the DELETE key to undo a mistake. When you are reasonably certain you typed the right password hit RETURN once. If there is a message about an invalid login, try again, starting with your user name. Otherwise, you have entered your first conversation (or, more professionally, *session*) with a computer. What a glorious day!

X Things are now going to happen! The screen changes and boxes appear out of nowhere. Sit back and wait half a minute or so for the action to stop. What you see depends on the setup of your account and can be very highly customized. Customizing is a delightful and gratifying process. It is also an advanced topic, and so we won't cover it here. Instead we'll assume you have the default version of the window system *X* going. If you discover discrepancies between this handout and what you see then somebody has altered the default setup without bothering to update the handout.

The Background There are a number of boxes on the screen that seem to be in front of something (called the *background*) that may be a uniform color (typically some gray raster pattern) or a fancy picture (like the full moon at the time of this writing), depending on the particular setup. The boxes are called *Windows.* Now you have to learn how to manipulate those windows.

[5] If you have a private workstation you may see just a single window. In that case log in and type `xstart` to get X going.

The Mouse Move the mouse around on its pad. Something (called the *pointer*) will move on the screen. You use it and the mouse buttons to manipulate the windows.

Icons Windows can be open or closed. If they are closed they turn into a very small box called an *icon*. Closing and opening a window is also called *iconizing* and *de-iconizing* it, respectively. To iconize a window move the pointer to the small box in the upper left corner of the window, push the left mouse button, and let it go. We'll call this *clicking* from now on. To deiconize the window move the pointer onto the icon, and click the left button again.

Moving a window To move a window move the pointer onto the grey bar at the top of the window, push the right hand button down, keep it down, and move the mouse on its pad until the window is in the desired place. Let go of the mouse. Try it now, using the window called *Local*!

Resizing a window To resize a window move the pointer onto the little box (the *resize box*) in the upper right corner of the window, push the left mouse button, keep it down, move the mouse until your window has the desired shape, and then let go of the mouse. Try it now! You can also make a window fill the entire screen by clicking with the left mouse button on the box to the left of the resize box. That box is the *zoom box*. To undo that effect click on the zoom box again.

Menus Next, move the pointer somewhere onto the background. In turn, click each mouse button and keep it down for a few seconds. Little windows will pop up that contain strange words. These are *Menus*. You give commands by moving the mouse (with the appropriate button still depressed) to the desired item and then letting go of the button. Let's try it. One of your menu items should read *clock*. (In my setup it's the second row in the menu popped up by the middle button.) Try that one and—presto—you should have a working clock in the upper right corner of the screen. Try the same thing with the line marked *Console*. A new window should pop up. Move it out of the way and go on. If you like play with other menu items. For example, you might like to figure out what the *zoom* items do.

The Icon Manager In the default setup, windows are represented redundantly on the screen. Eventually you will need to figure out what's best for you. For the time being I suggest you remove the redundancy and possible confusion by activating the left menu item called *Hide Iconmgr*. The box in the upper left corner of the screen should disappear. Congratulations! You just carried out your first and temporary—but very own—customization.

Creating a Window One way way to open a new window is to activate the menu item called *local*. It's line 4 in the menu going with the right button. Do it. This time the faint outline of a window appears. Move it around on the screen until you have it in the right place. Then push the left Mouse button.

Focusing One of your windows is going to be the one that receives whatever you choose to type on the keyboard. You select that window by moving the pointer into the main part of the window. No need to click anything! This is called *focusing* on the window.

Raising a Window If a window is partly obscured move the pointer on the grey bar on the top of the window and click the middle mouse button. The window becomes fully visible. This is called *raising* the window. (Similarly, you can *lower* a fully visible window. Figure this out for yourself.)

Scrolling a window To do this little exercise we need to fill up a window. So make sure your window called *Local* is fully visible and the pointer is in its central part. Type `pwd` followed by RETURN. Then type `ps -aux` followed by RETURN. Repeat this sequence two or three times. Do it now! You have actually just given some commands that the machine duly executed. Suppose you wish to see what the first command did. That was *pwd*, remember? Locate the top of the grey bar in the left margin of your window. Move the pointer on it. Push the middle mouse button, keep it down, and slowly move the mouse up as far as it will go. Let go of the mouse. You should see the contents of the window scroll by. At the top row you should see the word *pwd*, and in the next line a funny strings of words separated by slashes (/). (That's your "login directory", more about that later.) I'm assuming that you have not yet typed anything else in that window, otherwise the word *pwd* will occur earlier as you move up the mouse.

Removing a Window Many windows can be gotten rid of by focusing on them and typing the command `exit`. If that fails, and you don't know what else to to use the menu item *kill* (on the left button, about three quarters of the way down. Let go of the mouse button. The pointer turns into a skull. Move it onto the doomed window or icon, and click the left mouse button. If you change your mind and decide not to kill any window click the left mouse button on the background.

Cut and paste To transfer text from one window to another, say window 1 to window 2, do this:

— Move the pointer to the beginning of the region you want to cut in window 1. Press the left mouse button and keep it down.

— Move the pointer to the end of the region you want to cut in window 1. Release the mouse button.

— Move the pointer to the point in Window 2 where you wish to paste the text you just cut. Push the middle mouse button.

Caveat This does not work with windows running Emacs (see below) under the default setup. Personally, I consider this a major deficiency. That's the reason why my customized setup is different. In a week or so you may wish to explore customization yourself.

Exiting X Before we go on you need to know how to terminate your session. You do this by selecting the menu item *Exit X-windows* on the bottom of the left button menu.

Skip If you have worked your way patiently through to this point you now get to skip the next one, and continue with the one called *UNIX*. You see, hard work does lead to unexpected benefits!

Getting Down to Work

Coming in over the network I assume that if you are accessing our systems over the network from a remote location then somebody has told you how to do this[6]. But you do need to know the names of the hosts you may use. They are:

jeeves. A DECstation 5000 recommended for general computing purposes. Accessible only over the network.

solitude. A SPARCstation 1 dedicated for electronic mail, editing, and word processing. Accessible only over the network.

cayley. A SPARCstation 1 intended for jobs requiring a large amount of memory (40 Mbytes). Accessible only over the network.

[6] Typical access software is *telnet* or *Versaterm*, but there are other programs.

sandy and **magna.** Two Sun SLC stations available for general computing, accessible only over the network.

glab1 through **glab6.** 6 color DECstations 3100 located in JWB 120, to be used as consoles or over the network. JWB 120 is our "graduate lab" for use by faculty and graduate students. You can request a key and key code at the front desk.

uglab1 through **uglab30.** 30 monochrome DECstations 3100 located on the north side of the upper level in Building 129 (immediately East to the Math department), to be used as consoles or over the network.

graphics. An Ardent Titan graphics workstation, billed a "personal supercomputer" by the vendor, located in JWB 120. Graphics use takes precedence over all other uses on this machine. If not used for graphics purposes the machine can be accessed like any other over the network, or used as a console.

UNIX

There is now a conceptual (if not actual) break. We are done with a first introduction to the window system, and need to talk about the language used by your workstation. It's called *UNIX*. Before we start play with the window system some more if you wish. Then move the window called *Local* to the top and focus on it. For quite a while all the action will take place in this window. Unix functions independently of the Window system. For example you might call up one of our workstations via a modem. Then you would have just one window, but UNIX would be the same in that window, and everything below would apply.

Upper and lower case

It's important to realize that unlike some other operating systems, UNIX distinguishes between upper and lower case letters. Typing `ps -aux` will list the processes running on your workstation, `PS -aux` will only cause a message like *PS: Command not found.* to be displayed.

Commands

You interact with UNIX by typing commands. In your current window there should be a phrase like

36 uglab17%

followed by a little black box. The phrase is the *prompt* and the box is the *cursor*. Whatever you type will show up in the cursor's place, and the cursor will move. Some words will cause the computer to do things. These are called *commands*. You terminate them by hitting RETURN which signals the computer to decipher and execute your command. Let's try this: Type

echo 'This is a new world'

followed by RETURN. Aha! The machine echoed the phrase 'This is a new world'. A new world it is indeed! From now on, type RETURN after each command without specifically being asked to do so.

Changing the password

You need to be able to change your password, and in fact you should change it the first time you log in. To do this type `passwd`. The machine will ask for your old password. Type it. Then you will be asked twice (to guard against typing errors) to type the new password. Do so! The new password is effective immediately. If you forget it you won't be able to log in again. Nobody can retrieve your password, but specially privileged people can change it. Contact the Lab Assistant if you forget your password.

Exiting Sometimes you will want to get out of a certain program, without knowing how to accomplish this. Just try a few ways and see what works for a given situation. Try `exit`, `quit`, the RETURN key, *Control-C* (i.e, push the key labeled *Control*, keep it down, push the key labeled *C*, and let it go), Control-D, Control-Z, the *kill* menu item, or the *Exit* menu item.

Interactive Help Most of the following commands allow a substantial number of modifications. To find out about these use the command `man`, for example try

```
man man
man cat
man less
man ls
man cd
man mkdir
man rm
```

etc. The command will display one page of documentation at a time. To see the next page type a blank, to leave the documentation type `q`.

Electronic Mail To use electronic mail type `mm`. To find out what you can do in `mm` type question marks, `help`, and `help command-name`, e.g, `help read`, `help send`, `help reply`, `help headers`, `help help`.

Plotting Graphs There is a symbol manipulation language called *Maple* with numerous abilities, including that of plotting graphs of functions. To see this in action do the following:

— In the window called *local* type `xmaple` (followed by RETURN).

— Move the outlined box to an appropriate place and click the left mouse button.

— Focus on the new window (called *Maple V*) and type

```
plot(sin(x),x=0..6.28);
```

followed by RETURN. Make sure you type the semi-colon!

— Position the newly outlined window suitably and click the left mouse button. The window shows a plot of the sin function in the interval $[0, 6.28]$.

— In the window called *Maple V* type

```
plot3d(x^2+y^2,x=-1..1,y=-1..1);
```

Make sure you type the semi-colon and follow with RETURN. Another window appears. Click on its buttons and see what happens!

Symbol Manipulation In the window called *Maple V*, type the following:

```
diff(x**3,x);
int(x**3,x);
factor(6*x**2+18*x-24);
ifactor(120);
```

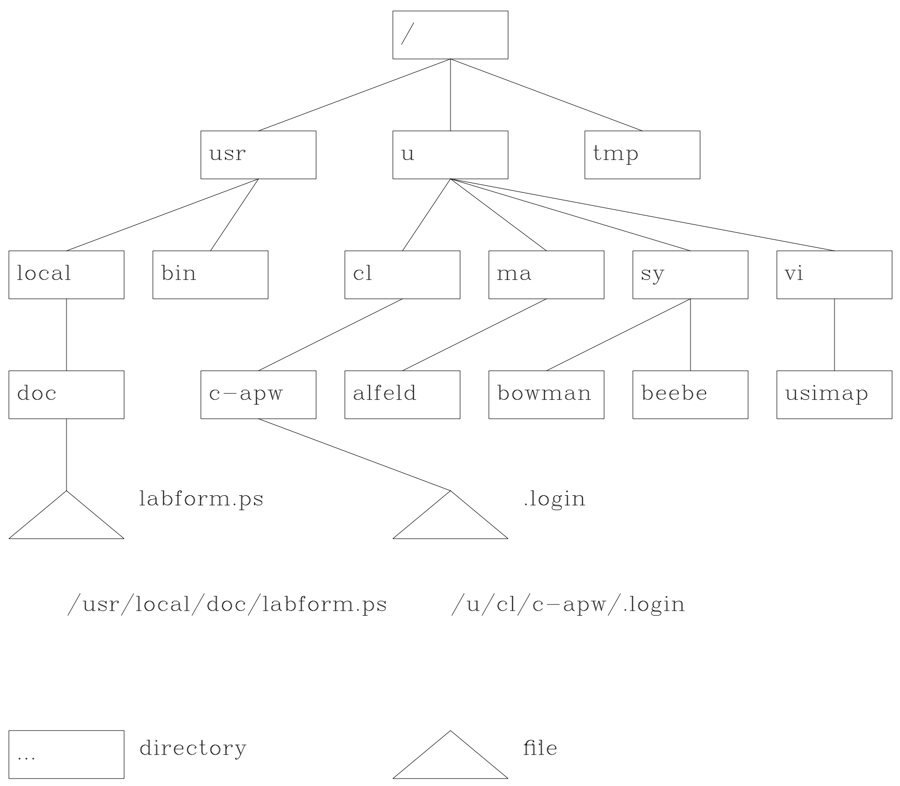

Figure 2. The Directory Tree.

```
?help
```

Files and Directories

A fundamental concept in Unix is that of a *file*. This is a collection of information, usually kept on a magnetic disk. A useful analog employed extensively on Macintosh computers is that of a folder. A folder can contain other folders, and a file can contain other files. When it does it is still a file[7], but usually it is called a *directory*. There is one directory that contains all others, directly or indirectly. It's called the *root directory* and denoted by a simple dash. The directories a directory contains are its *subdirectories* which in turn may contain other directories. In this manner you obtain a tree structure with the root directory at the root (naturally) and the proper files at the leaves. A small part of the directory (with the root on top) is shown in Figure 2. Two files (one containing the form used for setting up lab accounts) are shown with their position in the tree and also their full name (or, more properly, *path*).

[7] Get used to that idea! Many of the finer points of UNIX become startlingly clear if it is fully appreciated that directories (and a host of other things like commands and devices) are files.

Listing file contents `cat` is a UNIX command that causes the contents of a file to be displayed. Try typing

```
cat .xinitrc
```

.xinitrc[8] is a file that determines what you see on your screen when the window system starts up[9]. (Note that the period is part of the file name.) A more advanced command to view a file is `more`. It lets you view one page at a time. To see the next page type a blank. To quit examining a file type `q`. To see what else you can do type a question mark. `more` lets you go backwards as well as forward (type `b` to go back). Try

```
more .twmrc
```

Aha! That's the file defining your menus. How does the machine know in which directory these files reside? It looks in a specific directory called the *working directory.* When you first start your session this is a special directory called your *login directory.* That's the place where your setup files are, including .xinitrc and .twmrc.

Printing the working directory Type the command `pwd`. This stands for *print working directory.* You already did that earlier and you should see something like

/u/cl/c-demo   or   /home/alfred/a/cl/c-demo

Which of these two you actually see depends on which machine you login on. Each of these two names describes the same directory. This is an application of *links* as they are described later in this handout.

File names File names can also be specified by a complete `path` starting at the root directory. A path is a sequence of directory names separated by slashes. For example, consider the file

/usr/bin/pwd

This is the file[10] *pwd* in the directory *bin* which is a subdirectory of *usr* which in turn is a subdirectory of the root directory. File names can also be specified by just one word if they reside in the working directory or a by a partial path (starting at the current working directory). For example, if our working directory is */u/cl/c-demo* then */u/cl/c-demo/fname1* and *fname1* describe the same file. Similarly *notes/crashc.tex* and */u/cl/c-demo/notes/crashc.tex* describe the same file. In addition, here are some useful abbreviations: ˜ denotes the login directory, . the current working directory, and .. the parent directory.

Listing files To list the files in your working directory type

ls -lisa

This gives you lots of information about your files not all of which you will understand at this stage, but which will begin to make sense soon.

Changing the working directory After you login you can work with files in other directories. You can designate any directory your working directory by giving the command *cd*, followed by the directory name. This is called *connecting* to the new working directory.

---

8 If you don't have this file type `cat .login`.

9 So if you want X to start up differently you need to change that file.

10 Yes! You have actually found the file containing the (binary) code for the command `pwd`. *bin* stands for *binary.*

Creating a directory The command `mkdir` command creates a new subdirectory of your current working directory. Try this: with c-xxx being your login name[11] (or login name to be) type

cd
pwd
mkdir xxx
ls
cd xxx
pwd
mkdir xxx
ls
cd xxx
pwd
cd ../..
pwd

As you do all this think about what every command does and interpret the output it generates. Where do you end up? (in your top level directory!

Deleting a file if *fname* is the name of a file in your working directory you can remove it (delete it) by typing `rm fname`. You will be asked for confirmation.

Deleting a directory If *dname* is the name of a directory in your working directory you can remove it (delete it) by typing `rmdir dname`. Note that you have to empty it first (by deleting the files in it)[12].

Copying files Th command `cp` duplicates a file. If *fname1* is the name of an existing file, the command

cp fname1 fname2

duplicates the file *fname1* and names the copy *fname2*. Note that the file names may be directories, and that they may reside in different directories (in which case they will include partial of complete path descriptions).

Moving files The command `mv` works like `cp`, except that afterwards the file *fname1* no longer exists.

Emacs This is our most powerful and popular editor. It can be used to create and modify files. To find out how to use it invoke an interactive tutorial by following this procedure:

— Type `emacs`. If you are running X a shadow box will occur. Activate it with the left mouse button. (Depending on your setup emacs may

[11] We use your login name to ensure that directories created during a demo involving many people using the same account are distinct. If you login under you unique user name you can of course use any directory names you like.

[12] Can I interest you in some philosophy? We said earlier that a directory is a file is a file. So it seems strange that *rm* shouldn't work on a directory. And it seems a nuisance having to empty a directory before removing it. This apparently is a safety feature to protect yourself against accidental wholesale deletions of the fruits of your labor. Now that you have patiently followed to this point I'll tell you how to remove a directory and all its progeny. Type `rm -r dname`. This will delete all subdirectories, subsubdirectories, etc., and their contents, of *dname*. Here's another point: Our systems people are extremely cautious. They have modified the standard UNIX meaning of `rm` so that it will ask you before deleting anything. This can become tedious. You can avoid it by typing `rm -rf dname`.

start up directly in the window in which you issued the command. If you are not running a window system emacs will start up directly on the screen.)

— When the screen has changed press down the button labeled "Control", and keep it down.

— Push the button "H", and let it go.

— Let go of the "Control" button.

— Push and let go of the button "T".

— Follow the instructions that will appear on your screen[13]. Plan to spend about a day on this.

TeX To use TeX, generate a file ending with ".tex", e.g., `crashc.tex`, then type `tex crashc`. To preview the typeset copy on your workstation type `xdvi crashc`.

Printing To find out how to print your particular file from your particular location on a particularly convenient printer type `man printing` and find the appropriate command.

Programming To write and run a FORTRAN[14] program, generate file whose name ends with ".f", e.g., *prog.f* that contains the FORTRAN code. Then type `make prog`[15] to compile and link the program. Get rid of any errors. Then run the program by simply typing `prog`. Proceed similarly for C programs, except use a file *prog.c* instead of *prog.c*. The best reference on FORTRAN is: Michael Metcalf, *Effective FORTRAN 77*, Oxford Science Publications, Clarendon Press, 1985, ISBN 0-19-853709-3; the best reference on C is: Brian W. Kernighan and Dennis M. Ritchie, *The* **C** *Programming Language*, Prentice Hall, 1988, ISBN 0-13-110362-8.

Aliases You can define new names for commands using the alias command. For example, `alias d ls -lisa` will cause the machine to interpret the command `d` as `ls -lisa` from now on. You may recall that `rm` asks before deleting a file. This is accomplished by defining `rm` actually to be an alias for the original UNIX command `rm -i`. You can remove any alias by using the command `unalias` (so `unalias rm` will restore `rm` to its original meaning). `alias` by itself will list all aliases in use. Try it now, to see what else the staff have inflicted on us.

Taking commands from a file You can read and execute commands from a file using the command *source filename*.

Batch jobs You can let jobs run in the background by giving the appropriate command and terminating it with an ampersand (&). If you do so you should also include the command `nice`. For example if you have a program called *doit* then—once you are quite confident it will do what you want it to within no more than a couple of days—you can start it by giving the command

```
nice doit &
```

[13] Are you ready for a real idiosyncrasy? The Emacs tutorial asks you to identify what they call the *Meta key*. On most keyboards this is the key labeled *Escape*. However, if you are working on one of our DECstations you will soon discover that there is no such key. Instead, the Meta key is the one labeled F11! In general you can also obtain the effect of hitting *Escape* by typing Control-[.

[14] Other available languages include e.g., C, Pascal, REDUCE, Maple, Matlab.

[15] By the way, the command `make` is a nifty standard UNIX utility that makes managing large tasks and directories much easier. For example, by setting things up suitably a student might type the command `make thesis` to cause programs to be run and documents to be typeset and printed. Try `man make` and then ask a friend for some hints. Depending on demand we may give a lecture or two on `make`. I'm now working on a command `make career`. Once I get the bugs out I'll be able to give it and then take my family to Lake Powell for the summer.

However, if you have a computing task requiring substantial resources (CPU time, memory, or disk space) it is wise to consult with the staff (call 1-5257) about the best way to get your work done.

Setup files You can have certain commands executed every time you login, logout, or start a new "c-shell" (which for our purposes means a new window). These commands are in the files ~/.login, ~/.logout,~/.chsrc, respectively. Examine these files now!

Recent commands To obtain a list of your recently given commands type *history*. Then you can cut and paste complicated (or even simple) commands. This works particularly well after giving the command *history -h*.

Repeat commands Previously given commands can also be specified by typing `!x` where `x` is the string that started the command. For example, `!l` will repeat the last command that started with `l`, and `!ls` will repeat the last command that started with `ls`. Two exclamation marks (`!!`) will repeat the very last command.

Wild Cards When specifying a file name you can use an asterisk to describe all possible strings. For example, `rm -f *` will remove all files (except directories) in your working directories. `rm -f *.dat` will remove all files ending with ".dat".

File name recognition Hitting the ESCAPE[16] key while typing a filename will cause the system to complete the file name automatically to the extent possible.

Links An advanced form of referencing files are hard links and symbolic links. These allow you to bypass the tree structure of the Unix file system. Type `man ln` for more information.

Pipelining Another advanced and useful topic is pipelining. Read about it in a book or manual on Unix, or talk to a friend.

Output redirection Do the same for output redirection.

Access privileges and Security You can change access privileges to your files with the command `chmod`. In particular, if you wish to make sure that only you can access your files[17] type

```
cd; chmod 700 .
```

It's crucial that you include the period in your typing! Type `man chmod` for more information. A general word about security: basically your files are as secure as your password. Other than yourself, only four system staff members have full access to your files. They need this so they can do their jobs. In case you are wondering, I myself have the same access restrictions as everybody else and **cannot** read your protected files (including your mail box). Of course, given sufficient motivation and effort many people could gain access to your files by various devices, so you shouldn't keep truly confidential stuff on our systems (or, for that matter, on any other computer).

Disk Space It seems that no matter how much disk space we install we run out soon thereafter. In many places this problem is addressed by imposing charges or quotas. For many years, we

[16] see the footnote at the Emacs paragraph.

[17] Our default protection is that anyone can read almost all of your files. Exceptions are your mail box, and files in student directories. By default nobody else can write, delete, or otherwise modify your files. If you wish to share write access to some of your files with a specific individual, e.g., for the purpose of writing a joint paper, we can arrange this (send mail to `cscproblems`.)

have refrained from doing so in order to provide as pleasant and hassle free an environment as possible. However, this works only, if everybody cooperates by voluntarily freeing disk space whenever possible. The following points describe some facilities that help you control your disk use. Just make an effort to delete large unneeded files such as pictures, copies of software that is already installed for public use, postscript files, and the like.

Disk use To find out about the disk use of your working directory and its subdirectories type `du`.

Available disk space To find out about available disk space type `df`. The first time you do this have someone explain to you what you see.

Cleanup To get rid of garbage files wholesale use the command `cleanup`. Type the command without arguments to see what it can do. An effective and safe use is to connect to your login directory and type

cleanup -af

Large Temporary Files You can store large[18] temporary files, including directory trees, in the directory */tmp*. In fact you should do so if you work with such files. This directory resides on the local hard disk of each individual workstation and therefore is unique to the workstation. Putting files into it rather than your login directory bypasses the file server and reduces the load on the server. It also keeps disk space free on the server. However, the directory is completely emptied each time the machine reboots, so you can't use it for permanent files. If you fill up */tmp* completely your workstation becomes essentially inoperative, so guard against that eventuality. If it does occur delete some of your files in */tmp*. To find out how much disk space is available in /tmp use the command `df`.

Networking You can open a session with another machine by using `telnet` or `rlogin`. To copy files between our systems and others use `ftp`. This command is also useful to bypass file protections when working with a (cooperative) colleague. Finally, you can have a single command executed on another system (on which you have login privileges) with the command `rsh`.

How to get help Talk to the Lab Assistant, call 1-5257, or send a message to *cscproblems*.

Literature A great reference[19] on UNIX is: Brian W. Kernighan and Rob Pike, *The Unix Programming Environment*, Prentice Hall, 1984.

---

[18] small ones too, of course.

[19] I lent my own copy to someone and forgot who. The book is so excellent, and so useful, it was never returned!