CS/EE 181 UNIX Introduction

Staff
Department of Computer Science
California Institute of Technology
Pasadena, California, U.S.A.

September 19, 1996

CS/EE 181 UNIX Introduction

Designing VLSI chips, like writing computer programs, requires both planning and diligence. Part of what you'll be doing is digital system design, a process that takes place in the familiar realm of thought and imagination, and requires little more equipment than pencil and paper. However, laying out and checking the resulting design requires computer-aided design (CAD) and analysis tools and the computers that run them. Since you will be expected to complete your first chip in a remarkably short time, you are well advised to become familiar and comfortable with these computing systems and design tools.

In the hope of minimizing the frustration of trial-and -error and the need for unlearning the unproductive approaches that go with it, here is a short guide to our facilities, how to access them, how to do the basic necessities, and how to get more information. The best way to use this "crib sheet" is to try things as you read along. For more in-depth information, use the on-line manual `man` or check the 4.4BSD manuals, of which there should be at least two copies in the lab.

Machines Available

You will be using Intel Pentium workstations, of which we have 18 in the CS-VLSI lab, located in Booth 1 and Booth 7. The machines run a variant of Berkeley UNIX and are connected with a high-speed network. There are also some Sun workstations; we may need these if a lot of people need to use the lab at the same time or if we are unable to find an Intel version of some software. (This work is still in progress.) Each machine has a name, called its *hostname*.

The characteristics and hostnames of the machines are:

`viola`–`lute` (`vlsi-3`–`vlsi-20`)	Intels in Booth 1
`guitar`, `clarinet` (`vlsi-21`, `vlsi-22`)	Intels for remote service
`stun4a`–`stun4z`	Suns in Booth 7 and elsewhere

Try to stick with the Intels if possible; they are about ten times as fast as the older Suns! The Suns and Intels share all files transparently, making them almost entirely interchangeable. You can sit down at any available computer, regardless of which one you used during the previous session. If you have any problems getting started with the systems, see Chris Lee or Mika at once. If you have any problems with the software (even non-course-related) or other suggestions, don't hesitate to send a message to `pentium-bugs@cs.caltech.edu`. We depend on your input to keep the systems up-to-date!

Logging In

To log in, find a PC offering a `login:` prompt. In order to conserve electricity and the CRT phosphors, we encourage people to turn the monitors off when they are done, so you may first need to find the power switch for the display. However **please don't turn the Sun or Intel computers off.** Apart from the fact that the operating system gets confused if you do that, other people may be using them or may want to use them through the networks. Incidentally, if you wish to use the software through the network, please feel free to do so. We encourage people to use the two machines set up for that purpose, `guitar` and `clarinet`, but you are not limited to those. Please contact the TAs if you want source code for the software to compile on another system.

Anyway, once you find a `login:` prompt, type your *username*, followed of course by hitting the "return" key. Wait for the `password:` prompt, and type your *password*. Notice that to keep it secret, your password does not appear on the screen.

Each student is assigned a single account for all the CS classes they may be taking, except CS 1/2/3 for which we have a separate lab. Student accounts are retained for as long as they continue to be enrolled in CS classes. Each student account is distinct from whatever other account a student may have (for example, for sponsored research), and is financed by a special CS department fund.

Control Characters

Input to line-at-a-time programs, such as the default top-level interpreter, the C shell (`csh`), may be corrected with the following characters. Control characters are typed by using the *control* key on the keyboard like a shift key (e.g., holding the control key down while typing the character), and are denoted in this description by preceding the character with `^`.

```
DELETE erase one character at a time
^W     erase back to the last space or tab
^U     erase the whole line
^R     re-display a command line after many corrections
^S     pause the output so you can read it
^Q     resume output
^C     interrupt the program, get back to a command prompt
^D     end-of-file (EOF), means ‘‘that is all I had to say’’
```

Most programs will quietly finish up and terminate when you type `^D` . Many also have some kind of "quit" command or abbreviation. Notice that in UNIX, "quit" does not have the connotation of an ungraceful end that it has

in normal English; it is the normal way out. When a hasty retreat is desired, some programs (such as mail) offer an `exit` command. Many programs allow you to type "?" to find out what your options are at their prompt.

The Unix File System

The UNIX file system is tree-structured, meaning that any directory can contain other directories, as well as ordinary files, programs and links (pointers) to files in another part of the directory tree. Sub-directories can be nested to any depth.

Each process has a *working directory*, which is where filename interpretation starts. When you log in, the working directory of the command interpreter is your *login directory*. The command to display the path from the root directory to your working directory is:

```
pwd
```

Try this. You should get a response along the lines of `/ufs/students/`*username*, which is the path from the root of the directory tree to the login directory for your student account.

File and Directory Names

To specify a file in the working directory, just give its name. Names can be any number of characters up to some (ridiculously large) limit. Uppercase and lowercase are distinct in file and directory names. Almost any character can be used in file names, although characters with a special meaning to the command interpreter may have to be quoted.

Appending a `/` and another name, which can be done to any number of levels, means that the name before the `/` is expected to be a directory, and the name after the `/` is to be sought in that directory. For example:

```
homework/set1/UNNAMED.mag
```

means that a directory named `homework` will be sought in your working directory, and it will contain a directory named `set1`, which in turn contains a file named `UNNAMED.mag`.

Each directory contains two special entries:

```
. is that directory itself,
.. is that directory's parent directory.
```

For example:

```
./myprogram
../proj1/buffer.mag
```

The first example would be the same file or directory as `myprogram`. In the second example, the directory `proj1` is sought in the parent directory of the working directory.

As a special case, if there is no name before the first `/`, the traversal is begun at the root of the file system. Examples:

```
/ufs/students/cs181/.login
/usr/cad/lib/magic/scmos/28p23x34.mag
```

To use such file specifications would require that you know the path from the root of the file system. Instead, if the first character is `~`, the name following is sought as a login directory. The following filenames:

```
~cs181/.login
~cad/lib/magic/scmos/28p23x34.mag
```

are equivalent to the preceding pair, because `cs181` and `cad` happen to be login directories. Finally, a prefix of `~/` results in the traversal starting at your own login directory, whatever your current working directory may be.

Directories

The all-purpose command for displaying the contents of a directory is `ls`. `ls` with no arguments lists the contents of your working directory. `ls` with a directory name argument lists the content of that directory.

Many UNIX commands come with *options*, which are often distinguished by being preceded by a `-`. For example:

```
ls -a
```

will show otherwise invisible files and directories such as `.`, `..`, `.login` (described later), and others.

```
ls -F
```

displays files and directories so you can distinguish them and their types. A `/` after the name indicates a directory, a `*` an executable file (program), a `@` a link.

```
ls -l
```

provides a long (detailed) form of directory listings that includes the sizes, ownership, dates, and protection modes of the files and directories. Options typically precede file or directory names in UNIX commands. Options can usually be combined, for example `ls -al`. You could also type this as `ls -a -l`.

You might try what happens when you type `ls`,`ls -aF`,`ls -al`,`ls ..`,`ls /`. You can probably explain why when you type `ls ..` from your login directory, you get a list of the usernames of all the other students with accounts.

The commands for creating and removing directories are:

```
mkdir name       create empty directory
rmdir name       remove empty directory
```

You might try creating and removing some directories. What happens if you try to create a directory with a name that already exists?

You can change the working directory with the command

```
cd name
```

where `name` is a directory name. `cd` with no arguments sets the working directory back to your login directory.

You can create as many directories within your login directory as you like, and delete any file within it (even ones that you don't own, such as the ones that we will supply to start you out). To minimize clutter, the files and directories that you use for different courses, projects, and other activities should have their own directories.

Copying, Renaming, and Deleting Files

Sometimes when you re-use a file in a new project it is convenient to have the file appear in multiple directories. You can do this by creating a *link*:

```
ln filename linkname
```

If you want a separate copy to modify, use:

```
cp fromfile tofile
```

which copies `fromfile` to `tofile`. To rename a file or directory use:

```
mv oldname newname
```

Notice that the arguments of all of these commands first specify the existing file(s), and then the destination. If the destination is a directory, the file(s) will be linked/copied/moved into that directory. For example you could:

```
mkdir save
cp .login save
```

which would make a copy of your `.login` file in a directory called `save`, and the new file would be `save/.login`. That is, the effect is just like:

```
mkdir save
cp .login save/.login
```

You can remove (delete) files or links with:

```
rm filename
```

When you remove a file it's really gone, and the only way to get back a deleted file is to ask us to retrieve an old version from backup tapes (which are made every day, and kept for a few months). Nevertheless, to forestall disk space exhaustion, you should remove files that you no longer need. You can review your current disk usage with `du` and find out how much space is left with `df`. You can also keep your disk-space usage down by using `compress` (try `man` compress).

Filename Wildcard

Right after describing `rm` is a great place to mention the filename "wildcard," `*`. Lists of filenames can be easily generated with the wildcard feature of the command interpreter. For example:

```
ls -l buf*
```

generates a list of all filenames starting with the string `buf`. The list is substituted for `buf*` in the command line, and the `ls` command then displays the size, date, protection, and names of those files. What we (purposely) neglected to mention above is that many commands, such as `ls` and `rm` will accept a list of one or more filenames as arguments.

Of course, powerful commands with wildcards must be used with care. Suppose we were meant to type:

```
rm proj/buf*
```

but instead slipped and typed:

```
rm proj/buf *
```

The result would be to remove `proj/buf`, and then to remove `*`, all of the files in the working directory! ARGH!! They who hesitate before hitting return are often saved.

Finding Documentation

At this point it gets a bit tedious to describe all the variants of the more complicated programs. To find out more about the relatively more complicated programs mentioned below, such as `rcp` (remote copy), `ftp` (file transfer program), `chmod` (change mode), the `csh` (C shell) command interpreter, and others, you should read through the corresponding manual entries. Printed manuals can be found scattered through 1 and 7 Booth, but the same or more current information is available on-line via the command:

```
man program
```

Most of the on-line documentation is concise; try man `pwd`. Some is long; try `man csh`, or `man mail`.

The problem with this kind of on-line documentation is similar to looking up in a dictionary how to spell a word. How do you find it if you can't spell it? If you don't know the name of the program that does what you want, try:

```
man -k keyword
```

to search through the section headings or "on-line" descriptions of programs. To get the "one-liner" of a program, you can type:

```
whatis program
```

Most of the standard UNIX programs are also documented in the 4.4BSD manuals spread in the lab.

Remote Logins

You can log into one machine from another with the `rlogin` (remote login) command. For example, if you were logged into `stun4a` and wanted to run a program on `stun4b`, you can type:

```
rlogin guitar
```

and you would find yourself logged into `guitar`. For remote logins to other machines, you will be asked for your password unless you have a suitable entry in a `.rhosts` file in your login directory (of the target machine). Similarly you can execute a single command on another machine with `rsh` if protections allow.

Pretty soon you will discover that there are a lot more machines on the network than those in Booth 1 and 7. *These other machines are off-limits unless you have specific permission to use them.*

File Protection

Usually only the file's owner (the user who created it) can modify it, but anyone can read it. The protection can be changed for individual files with `chmod`, or the `umask` command can be used to change the default protection for all files created by that process. The default allows read access to all so that you might have more opportunities to learn and share, this being a university. However, if

anyone complains that you are snooping on them, you will be fully responsible for your actions.

If you want to change file protections from the defaults and can't figure out the `man` page description, chapter 2 in the printed manual "Doing More with UNIX: Beginner's Guide'," is quite readable. Copies are scattered around the lab.

Redirection

Most programs take input (if needed) from the keyboard, and produce their output on your screen. The command interpreter can *redirect* the input or output to be any file, for example:

```
cifsum < myproject.cif
```

takes the input to the CIF checksum program from the file `myproject.cif`.

The program `cat` concatenates a group of files, and its output would normally appear on your screen, but its output can also be redirected, for example:

```
cat file1 file2 file3 > outputfile
```

A variant of output redirection uses `>>` to append the output to a file.

You can also connect the output of one program to the input of another, for example:

```
ls |  wc
```

runs the `ls` program to produce a list of filenames, which becomes the input for `wc` (word count) which counts them. In this way, simple programs can be combined to perform more complex functions.

There are many more handy things that the command interpreter can do for you, for which you are encouraged to run `man csh` or read chapters 3 and 4 in "Doing More with UNIX: Beginners Guide."

Your Search Path

You may wonder where these programs come from, once you don't find any of them in your own directory. The command interpreter has a *search path*, a list of directories that it tries to prefix to command names to form program

names. The search path is a *shell variable* (called path) set in an initialization file, `.login` in your login directory. Initialization files tend to be distracting, so their names all start with a period, and the `ls` program doesn't bother to show them unless you use the `-a` or `-A` options. We have supplied you with a few such files or links `-`, `.login`, `.cshrc`, `.xinitrc`, `.Xdefaults` to set up things like the search path, prompt and the window configuration on the workstations. When you feel ready, you can modify them or create your own.

Text Editors

All CS/EE 181 students need to know how to edit text. The Intels and Suns have an interactive tutorial programs for the `emacs` full-screen editor. The tutorial is invoked by running `emacs` and entering the help facility as directed. There are other editors available as well, `vi` is another full screen editor, and `ed` a line editor, are two examples. The `man` pages provide more information.

Displaying and Printing Files

Once you can edit files, you will likely want to be able to display them or parts of them, and to print them. The usual programs for displaying files on a terminal or workstation window are `more` or `less`, programs that have facilities for searching and many other things. They are also frequently used to paginate other output, e.g., you might try `man csh | more`. The programs `head` and `tail` are marginally useful for looking at the first or last few lines of a file.

To print text on the laserpinter in Booth 3 use:

```
lpr file
```

There are also a number of commands related to `lpr` for checking the print queue (`lpq`) or for removing items from the print queues (`lprm`). `ipr` is another text-printing program. As a useful example of piping the output of one program to another, you can print `man` pages on the printer by:

```
man program | lpr -L1 -p12 -s2 -T1 -126
```

Please use paper judiciously. Save some trees!

Communication Commands

```
mail            e-mail program (read the man page for details)
```

```
elm             fancier e-mail program
send,inc,scan RAND/UCI MH system (see above)
write type to another user almost in real-time
talk            type to another user in real-time
```

Miscellaneous Commands

```
clear           clears the screen
hostname        tells you the hostname of the machine you are on
date            gives you the time and date
leave           sets an alarm clock reminding you when to leave

jobs            list what programs are currently running
who             list who is logged on to your machine
ps -aux         list everything your machine is doing
```

X Window System

The X Window System, generally known as "X" or "X11", is the windowing interface we now use. X pretty much speaks for itself. The main thing you have to do to get used to it is to press the different mouse buttons when the cursor is in different parts of the screen, i.e., the background, in a window, in the window's control bar, or its corners. Pressing the control key while pressing a mouse button within a window provides extra interesting alternatives.

You can run any program in the background by appending a `&` to the end of the command and its parameters. This is most useful when the program pops up in its own window and you still want to use your original one. Try "`netscape http://www.cs.caltech.edu/~cs181 &`" at a prompt.

Other Things

Almost limitless. The Intels and Suns have compilers or interpreters for numerous programming languages, the TEXand LATEXtypesetting systems, and much more.

Integrated Circuit Design Tools

```
cadman          shows documentation for VLSI CAD programs
cadman -k       lists VLSI CAD tools relevant to keyword
magic           our main design tool (for this term)
plamin          switching function minimization program
```

```
mpla            PLA layout generation program
cifp            CIF plotter
cifsum          generates CIF check sums for MOSIS
magplot         plots magic files on the laserprinter
cosmos          MOS switch level simulator
irsim  another MOS switch level simulator
prsim yet another MOS switch level simulator
hspice          transient circuit simulator
```

and there are more that we will tell you about later.

To run `magic` to do the tutorials on any of the computers, you type in an `xterm` window:

```
magic tut1
```

But before you start `magic`, now take a minute to ensure that you can log in and start X on both the Intels and the Suns; you will need to be able to do this during the `magic` tutorial session during the second week of class.