Hitchhiker's Guide
to
VLSI Design with Cadence & Synopsys

David Money Harris
17 January 2009

The VLSI design tools at Harvey Mudd College are hosted on a Linux server named *chips*. This document introduces you to connecting with the server, getting around Linux, and starting the tools.

Connecting to Chips

The usual way to connect to chips is from your Windows, Mac, or Linux-based computer using the X11 display protocol. From Windows, you will need a program such as Cygwin to open an X terminal. Cygwin is installed on the ECF computers and you can download it for free onto your personal computer using the directions at

http://www4.hmc.edu:8001/Engineering/ClayWolkin/memos/CW08104C.pdf

If you have a Mac or Linux, the X Windows software is already built in.

If you are on Windows, choose Start • Programs • Cygwin-X • XWin Server. If you are on Mac, run /Applications/Utilities/X11.app. If you are on Linux, open a new terminal.

At the terminal, type `ssh -Y username@chips.eng.hmc.edu`, where you supply your username (usually first initial and last name, such as bkeller for Ben Keller). Chips will prompt you for your password.

The first time you log in, you'll be prompted for your year. A link to your Charlie account will be automatically set up based on that information.

You will need some files for the various labs. They are located in `/courses/e158/10`

Getting around Linux

Chips runs Red Hat Enterprise Linux Version 4, which is the old, stable, and preferred release for the CAD tools. If you aren't already familiar with Linux, follow this tutorial to learn your away around.

You can create, view, and manipulate files by typing commands into the terminal. Type

```
ls
```

to **l**i**s**t the contents of your directory. You'll initially see "charlie" as the only entry. We'll return to this special entry later. To create a new subdirectory, type

```
mkdir IC_CAD
```

when you ls again, you'll see

```
charlie            IC_CAD
```

To create a new file, type

```
gedit foo
```

The GNOME text editor opens up. Type "Hello" into the text editor, then save and quit. Now,

Typing

```
ls –l
```

Gives you a longer listing showing more information about the files:

```
drwxr-xr-x  2 bkeller users 4096 Jan 29  2009  charlie
-rw-rw----  1 bkeller users    6 Jan 17 12:54  foo
drwxr-xr-x  2 bkeller users 4096 Jan 17 12:44  IC_CAD
```

The owner is **bkeller**, who is part of the group **users**. The d indicates that both files are directories. It is followed by three triples of letters. The first wxr means that the user can **w**rite, **r**ead, and e**x**ecute the file (for directories, execute means to enter the directory). The second means that members of the same group can read and execute but not write the file. The third means that others not in the same group can also read and execute but not write. The file size is 4096 bytes, which is standard for a directory. The IC_CAD directory was created on January 17 of this year at 12:44. foo is not a directory and is only 6 bytes long (five characters plus the carriage return at the end). It is readable and writable only by bkeller and other group members.

In Linux, file names beginning with a dot are normally not displayed when you type ls. Usually, configuration files are named this way so they don't clutter things up. To display everything including the dot files, type

```
ls -al
```

Some dot files have already been created in your directory. For example, .bash_profile is a configuration file invoked on log-in that sets up the paths to find various programs.

To **p**rint your current **w**orking **d**irectory, type

```
pwd
```

And you'll see

```
/home/bkeller
```

To **c**hange **d**irectory into the new directory you created, type

```
cd IC_CAD
```

Now if you type `pwd`, you'll see

```
/home/bkeller/IC_CAD
```

To go back up to your home directory (`/home/bkeller`), type

```
cd ..
```

Where .. indicates the directory above where you currently are. (Note that . refers to the directory where you currently are.) Alternatively, you could have typed

```
cd /home/bkeller
```

or

```
cd /home
cd bkeller
```

or

```
cd ~bkeller
```

where `~bkeller` is a shortcut for the home directory of that user (i.e. `/home/bkeller`)

or just

```
cd
```

because `cd` returns you to your home directory from wherever you are.

To view the contents of a text file, you can open it in an editor. However, for short files, it is often more convenient to use the `more` command.

```
more foo
```

To **m**o**v**e or rename a file, type

```
mv foo foobar
```

To **co**p**y** a file, type

```
cp foobar fooseball
```

To **re**m**o**ve a file, type

```
rm foobar
```

Then type `ls` to see

```
charlie fooseball IC_CAD
```

Linux supports symbolic **li**n**k**s, in which one file is really just a pointer to a second. If you change the second file, the first will change. For example

```
ln –s fooseball foolink
```

Then when you ls –l, you'll see what foolink links to. If you were to use gedit to change fooseball to say "Goodbye", then typed more foolink, you'd see Goodbye. Thus, symbolic links are better than copying when you want two files to remain the same.

To **ch**ange the permissions on a file, use `chmod`. `Chmod` takes three octal numbers, indicating the permissions for the user, the group, and for others. Each number should be the sum of three bit fields: 4 for read, 2 for write, and 1 for execute. For example,

```
chmod 644 fooseball
```

changes the file to be readable and writable by the user and read-only for everyone else.

Then you could move `fooseball` into `IC_CAD` and rename it back to `foo`

```
mv fooseball IC_CAD/foo
```

To get rid of the entire `IC_CAD` folder and its contents, use the recursive option on `rm`:

```
rm –r IC_CAD
```

Use this command with care!

Open `gedit` again by typing

```
gedit
```

While it is running, the terminal is tied up and can't be used. To fix this, press ctrl-Z in the terminal to put `gedit` to sleep. Now `gedit` becomes unresponsive. Then type

```
bg
```

In the terminal to run gedit in the **b**ack**g**round. Now, you can use both gedit and the terminal. To find out what **p**rocesse**s** you have running, type

```
ps
```

And you'll see something like

```
  PID TTY          TIME CMD
29382 pts/1    00:00:00 bash
29905 pts/1    00:00:00 gedit
29949 pts/1    00:00:00 ps
```

bash is the "**B**ourne-Again **S**hell," part of the terminal that interprets the commands you have been typing. Gedit is running in the background. ps is the command you just invoked. Notice that each command has a process ID (PID). If a program locks up, you can kill it based on its PID. For example

```
kill 29905
```

kills the gedit session. You would lose any unsaved work, so don't use this unless you have to.

Instead of putting gedit to sleep and restarting it in the background with separate commands, you could have typed

```
gedit &
```

To invoke it in the background in the first place.

Your chips account should be automatically linked to your chips account. To access your account, change into that subdirectory. You can then copy files between the systems.

```
cd ~/charlie
```

In the Charlie folder is a README file. To read it and learn about changing your password, type

```
more README
```

Another way to move files between your computer and chips is through an **f**ile **t**ransport **p**rotocol (FPT) client. WinSCP is a good FPT program for Windows.

To get more information about a command, use `man`:

```
man rm
```

Moving to the Next Level

The commands so far are adequate to get you started. However, Linux is full of other commands that you will find useful as you become more sophisticated and work with a greater assortment of files.

`!!`: repeat the last command
`!word`: repeat the last command that started with word

`*`: matches any file name
Ex: `rm *.txt` removes all files ending with the suffix .txt (dangerous!)

`grep`: global regular expression print
Ex: `grep Hello *` prints the lines of all files that contain the word "Hello"

`|`: pipe the output of one command to another
Ex: `grep Hello * | more` displays the results of the grep with more

Other commands

`gpdf &`:	read a PDF file
ps2pdf:	convert a Postscript (.ps or .eps) file to PDF
`xterm &`:	create another terminal window
`exit`:	closes a terminal
`top`:	display a list of the processes using the most resources

A longer list of commands is at http://www.ss64.com/bash/

Running Linux and Windows Together

If you find you like Linux and want to install it on your computer but still need access to Windows-based software, you might want to run it alongside Windows on your PC. You can configure your computer as dual-boot and select the option of running either one at startup, but this is a pain because you have to reboot to switch. If your computer has plenty of disk and RAM, a better option is to use virtualization to run Linux and Windows side-by-side. You can do this by installing the free Microsoft Virtual PC software, then installing Linux on the virtual PC. For more information, see

http://www.pcreview.co.uk/articles/Windows/Run_Linux_in_Windows/