

Using Linux

You will do your work for Comp11 with the Linux computers at Halligan Hall. You can use the workstations in the labs, and you can connect to the servers over the Internet. This lab will show the basic ideas and skills of using a Linux system. Linux can be used in two ways -- as a graphical desktop and also using a text-based terminal window. When you connect over the Internet, you usually use the text based terminal mode. Today we shall introduce you to both ways of using Linux.

☞ Login now with your utln and CS password.

Graphical Desktop

When you login to a computer in the lab, you see a standard graphical desktop. It is not exactly Windows, nor is it exactly like the Mac OSX desktop, but it has most of the same stuff -- icons, menus, a clock, a mouse cursor, stuff you are probably familiar with.

- ☞ Explore some menus to see what they contain
- ☞ Make a new folder by right-clicking on the desktop

Text-Based Connection: History, Purpose

Unix was invented before PCs, Macs, and graphical desktops. People connected to computers over telephone lines and simple wires. These people typed text commands and got text answers. Much of the power of Unix/Linux is available from the text-mode interface. When you connect over the Internet using secureShell, ssh, or PuTTY, you will use a text-mode interface, so we shall spend most of this lab getting familiar with this way of using Unix/Linux.

Opening a Terminal Window

Under the Applications Menu, find the Accessories item, and then find *terminal*. Click on that. You will see a text-window containing a menu bar at the top and then a blinking cursor next to a short message.

This short message is called the *prompt* because it is how Linux prompts you to tell it what you want to do.

☞ Type the following commands:

```
who
date
uptime
cal
cal 2012
cal 2 2012
cal 2 12
```

☞ Use the cal command to find out on what day of the week you were born.

Right now, change your prompt so it is more useful.

☞ Do this:

Type: /comp/11/files/setup

Now type: exit

Now open a terminal again.

Remarks on Unix/Linux Commands

The things you just typed are called *commands*. These are names of programs. Nowadays, people refer to programs by the term *Apps*. We shall refer to these programs as commands, programs, or tools.

command : the name of the program, e.g. cal

arguments : The extra information you give to specify what the command should do. For example, the number 2012 tells the cal tool to print the calendar for the year 2012. If you do not specify arguments, the command will perform its *default action*. It's like ordering a coffee without specifying how much cream or sugar -- you get the default settings.

There are hundreds of Unix/Linux commands included with every version of the operating system. Most of these tools do not appear on the drop-down menus on the desktop; you have to know how to find them and how to use them.

The Unix Manual

All the Unix commands are explained in the on-line help system, called the *manual*. To learn about the cal tool, for example, use the **man** tool:

☞ man cal

which displays the help page for the cal tool. These help pages can be terse, but have all the information about the command. Experiment with some of the various options offered by this tool.

What if you do not know if there is a tool for a task? How can you search the manual to find what you need? The man command has a keyword search feature using the *-k* option. For example:

☞ man -k calendar

lists all manual pages that have the word "calendar" in the summary. Try this:

☞ man -k fun

which tells you more than you probably want.

Connecting to Servers

You can use the computers in Halligan from anywhere on the Internet. You just need a *secure shell* program. Mac and Linux machines include one; we have one

available for Windows machines.

Connect to the linux server here at Halligan by typing

```
☞ ssh utln@linux.cs.tufts.edu
```

Now type

```
hostname
who
finger
last -10
exit
```

You can even log into machines here in the lab. For example, we can all login to lab116F here in the lab with

```
☞ ssh utln@lab116F.cs.tufts.edu
```

Files and Directories

You will use the Linux computers to write programs. These programs will be stored in folders in your account on the server. We shall now look at the basic commands for managing files and folders.

Creating New Folders

In Linux, a folder is called a *directory*. You can call it a folder, but the natives use the word directory.

☞ Try this:

```
cd
pwd
cd Desktop
mkdir folder1
mv folder1 stuff
ls -l
```

As you see, these Unix commands create and rename a folder on your desktop. In fact, the point and click operations you perform use the same internal mechanism that the terminal commands use.

Now, using Unix commands, not the mouse, do this:

- ☞ make a Desktop folder called comp11
- ☞ in that folder, make a directory called lab2
- ☞ In Graphics Mode, click to open the lab2 folder

C++ Programming: Creating New Commands

Where do commands come from? People write them in C or C++. They write a program. Then they compile the program to get a new file: the command. We shall now convert a C++ program into a command. The sequence of steps to build a new command is:

```
Write
Edit <-----+
Save
Compile
Run
Test
Debug, Improve ----+
```

This is the pattern of work you will follow all term. The program is already written, so we move to the

Edit step. We shall first use a text-mode editor.

First, use **cd** to get yourself into the comp11/lab2 directory. Type **pwd** to make sure you are there.

Now copy the class example to your current directory:

```
cp /comp11/files/lab2/avg2.cpp avg2.cpp
```

And edit the file with the kate editor:

```
☞ emacs -nw avg2.cpp
```

☞ Put your name and the date in the comments at the top. Save the file by typing Ctrl-X then Ctrl-S. Then press Ctrl-X then Ctrl-C to close the editor.

Now compile the program by typing:

```
g++ -Wall avg2.cpp
```

This will produce a new command called `a.out`. You should see the new program appear in the graphical window of the directory. In text mode, you can run this new command by typing

```
./a.out
```

Congratulations! That is your first C++ program, written and compiled.

Bugs

We discussed syntax errors and logic errors in the last lab. Open the program in the editor and do this:

- ☞ Add one syntax error and add one logic error.
- ☞ Save the file
- ☞ Compile it
- ☞ Correct the syntax error
- ☞ Compile it
- ☞ Run the program

The program still has a logic error. To fix it, we shall learn to edit program over a remote connection.

Run-Time Errors

There is a third kind of error: runtime errors. These errors happen when the program is running. One cause is unexpected user input. Try this:

☞ Run the program, enter bad input, for example, enter the value 4.5 instead of the 4. What does the program do?

Correct/Simplify/Clarify

Here are discussion questions:

1. Change the program to handle doubles.
2. Change the program to average four doubles.
3. How can we handle bad input like *four* for 4?
4. How can we average any number of values?