Week 2 Lab

Intro to Linux

This is a very long lab sheet because it contains a lot of material to help you use Linux. Feel free to work with other students in your lab, and to finish it outside class if you do not have time to finish in class. Other labs this year will assume that you know how to do all the things described here, including installing new software, managing files, and operating in a virtual environment.

Task 1. Setting up your environment

Before you do any other work in this unit, you will need to set up a Linux development environment that you can run in labs. You will be able to complete some of these tasks using the Linux environment, but some of the things you'll be doing, such as Task 7, will require you to have root (i.e. administrator) access. You do not have this level of privilege on the lab computers.

The Linux distribution we recommend is xubuntu, and this is the distribution that will be used in lectures. If you prefer another distribution, you may use that instead. **Bear in mind that, for our purposes, MacOSX is *not* a distribution of Linux.** It is a version of Unix, but it's derived from BSD and has had many changes made to it along the way. If you develop under OSX and your code doesn't compile or run under xubuntu, you'll fail the assignments.

You have three options:

1. You can use a *virtual machine* on any lab computer that has VirtualBox or VMWare installed. We recommend you store it on a USB stick. You'll need at least 8Gb, and although any drive will work, it'll run much faster if it supports USB3. You can download a pre-made xubuntu VM image from `http://www.osboxes.org/xubuntu/`, or if you prefer, you can download an xubuntu installer from `https://xubuntu.org/getxubuntu/` and install it to a new virtual machine by following the instructions at `https://www.virtualbox.org/manual/ch01.html#gui-createvm` (for VirtualBox) or `https://kb.vmware.com/kb/2013483` (for VMWare Player).

   If you are using a Linux virtual machine for any other unit, *and you have the root password for it*, feel free to use that VM for FIT3042 as well.

   Note that you can still use your virtual machine in native Linux labs. VMWare Player is installed there and will happily run your VMWare or VirtualBox image.

2. You can use a virtual machine on your own laptop and run it there. This will be faster than running it from a USB stick, but not as fast as booting Linux natively. You can use VirtualBox, VMWare Player, or any other virtualization software.

3. If you have your own laptop, and you don't mind giving up some of your disk space, you can install Linux natively. The distribution we're using is xubuntu (`http://xubuntu.org`) but most distributions ought to work. This approach will be fastest of all, but is also the most technically demanding. You should be aware that installing a new operating system on a computer will usually require reformatting the hard drive, which will delete all files and programs that were already there.

   If you are already running Linux on your laptop, that's fine as long as you know how to bring up a terminal and either know the root password or have enough privilege to be able to use `sudo`.

   Your demonstrator can support xubuntu to an extent, but if you choose to use a different distribution then we do not guarantee that we'll be able to fix it for you if it breaks.

Before you proceed with the rest of the lab sheet, you will need to start a Linux session and log in.

Task 2. Getting started in xubuntu

Open a terminal. In xubuntu, you do this by clicking the icon at the top left of the screen.

xubuntu's "start button"

Look for the menu item labelled **Terminal Emulator** and click it. This should open a terminal.

Task 3. Getting help

As we saw in lectures, the main way of getting information about Unix commands is to use the `man` command.

The `man` command has several different sections. You can get a list of what's in them by typing `man man`, but the most important sections for this unit are:

- section 1, which documents programs and shell commands;
- section 2, which documents kernel functions that you can call from C programs; and
- section 3, which documents library functions

Exercises

a) Read the manual page for the `man` command and find out how to specify the section of the page.

b) Display the manual page for the `printf` program (`/usr/bin/printf`)

c) Display the manual page for the C function called `printf()`

d) What program can be used to alter the colour settings for the `ls` command? Display its manual page.

Task 4. Using bash

Now, we'll take a look at `bash` – the shell program that responds to your terminal inputs. Here are a few tips that can make it easier for you to use `bash`.

4.1 **Tab completion** – if you type the first few letters of a command or filename and then press the Tab key, `bash` will automatically complete the filename. If the letters you put in are not a unique prefix, then pressing Tab once won't do anything but pressing Tab a second time will bring up a list of all possible matches.

Tab completion saves a lot of time and effort, especially if you are working on a program that has long filenames.

4.2 **Setting and viewing environment variables** – many aspects of `bash`'s behaviour, and the behaviour of other Unix programs, can be controlled by environment variables. These variables are maintained by the shell, and can be set and queried at the command line.

Later on, we'll see how you can use shell environment variables inside your C programs, to allow users to change the way your program behaves at run time without needing to use a configuration file.

To set a variable in `bash`, simply type *`VARNAME=value`*. For example:

```
FOO=bar
```

If your value needs to contain spaces, you can surround it with quotation marks (''):

```
FOO2="more than one word"
```

You can view the contents of a variable, use the `echo` command:

```
echo $FOO
```

4.3 **`sudo`** – this program allows you to run programs with the privileges of another user. Since you'll probably be the only real user on your VM, you'll almost certainly use `sudo` to access root (i.e. administrator) privileges.

Unix-based systems such as Linux usually don't allow ordinary users to modify important files and directories. On multiuser computers, this prevents people from accessing one another's private data. Your user account on the VM has been granted the ability to use `sudo` so that you can do administrative tasks when you need to while protecting the system against accidental damage.

To use `sudo`, simply type it at the start of the command that you wish to run with higher privilege. For example:

```
sudo cat /etc/shadow
```

The shell will prompt you for your password – type it in. (If you're using the FIT VM, your password is xubuntu.) Your command will now execute.

Exercises

a) Use tab completion to find a program beginning with the letters "ca" that can tell you interesting things about today's date. Run this program.

b) Are `bash` environment variable names case-sensitive? Conduct an experiment to find out.

c) What happens if you leave the dollar sign out when you are using `echo` to display the value of a variable? What if you include the dollar sign when you are setting the variable?

d) What happens if you try to `echo` a variable that has not been defined?

e) Try to run the command `cat /etc/shadow` both with and without `sudo`[1]. Familiarize yourself with the error message that is displayed when you try to read a file that you don't have read permission for.

f) Your *prompt* – the text that is displayed when `bash` is ready to accept your input – is controlled by a variable called `$PS1`. Change this variable to make it include your name[2].

Task 5. Dealing with archives

As is the case with Windows and OSX, programs intended for use under Linux are usually distributed in archive form – that is, bundled together and compressed. The most common compression utilities used for this are `zip`, `tar`, and `gzip`.

You are probably familiar with zip archives from other operating systems you have used. To decompress a zip archive on the Linux command line, you use the `unzip` utility. For example:

```
unzip myarchive.zip
```

The other common format, `.tar.gz` or equivalently `.tgz`, uses a combination of two programs: `tar`, which is short for *tape archive*, and the compression program `gzip`. The `tar` program con-

[1] The file `/etc/shadow` contains encrypted passwords, so on a multiuser system it's a real security risk if non-administrators are allowed to read it. Recall from lectures that multiuser capability is embedded deep in Linux's DNA.

[2] If you'd like to experiment with $PS1, look at the section of the `bash` manual page entitled "Prompting". Don't worry – you won't damage the system by setting $PS1 at the command line. If you make the system hard to use, you can simply close the terminal you are using and open another one.

catenates the input files into one long sequence of bytes, as you would if you were going to back them up onto a magnetic tape drive, and `gzip` then compresses that sequence.

This format is often used to distribute source code for Linux and Unix programs. Source code repositories that are packaged this way are called *tarballs*.

You can decompress and unpack a tarball all in one go using the `tar` command. For example:

```
tar zxvf myarchive.tar.gz
```

a) Download the file called `files.zip` which you will find on Moodle to your Linux home directory. Unzip it on the command line.

b) Download the file called `morefiles.tar.gz` from Moodle to your Linux home directory. Decompress it on the commend line.

c) Look at the manual page for the `tar` command. What do the flags `zxvf` mean? Which flags would you use if you wanted to *create* a tarball rather than extracting one?

Task 6. File management

This task requires you to have completed the exercises from Task 5.

Although Linux has a GUI-based file explorer similar to Windows Explorer or the OSX Finder, it is usually faster to do simple file manipulations on the command line. Moreover, when we start writing `bash` scripts later in semester, it will definitely be to your advantage to know how these file commands work!

The exercises in this task will require you to use the following commands:

- `cd` – change working directory

  Examples: `cd subdir`; `cd ..`; `cd /home/xubuntu/files`

- `ls` – list files

  Examples: `ls`; `ls subdir`; `ls *.txt`

- `cp` – copy files or directories

  Examples: `cp file /path/to/new/location`; `cp -r directory newdirectory`

- `mv` – move or rename files

  Examples: `mv oldname newname`; `mv filename subdir`

- `mkdir` – make a directory

  Example: `mkdir newdir`

- `rmdir` – remove a directory

  Example: `rmdir newdir`

- `rm` – delete files

You'll also need to be aware of a few special names that you can use on Unix command lines. Double dot (`..`) as a directory name or path means "the parent of the current directory". Single dot means the current directory. The star (`*`) is a simple regular expression that matches any sequence of characters. So, for example, `ls c*` will list all files whose name begins with a c, and `ls *.c` will list all files whose names end in .c (i.e. C source files).

Exercises

a) While still in your home directory, list all the files in the `ProjectGutenberg/books/fiction` directory.

b) Use `cd` to move to the directory containing books by Jane Austen. List all the text files there (and only the text files).

c) This directory contains both fiction and nonfiction books. Use `cp` to copy each of the fiction books to the `ProjectGutenberg/books/fiction` directory, and copy the nonfiction books to the `ProjectGutenberg/books/nonfiction` directory.

d) Move up to the parent directory and try to use `rm` to delete the `JaneAusten` directory. Does it work? If not, how do you think you might delete this directory?

e) Look at the manual page for `cp` to find the option that makes it to a *recursive* copy – that is, to copy all files and subdirectories inside a directory. Go back to your home directory and use this option to copy the entire `JamesCook` directory into the `ProjectGutenberg/books` directory.

f) In your home directory, type `rm -ri books`. Observe what happens. What does the `r` option do? What does the `i` option do?

g) Create a new directory under your Documents directory called `Gutenberg Books`. (Note that you will need to put "double quotes"around the directory name in order to get an embedded space – otherwise you'll make two directories, one called `Gutenberg` and one called `books`). Copy all the books in your `ProjectGutenberg` directory to the new directory. Delete the `ProjectGutenberg` directory.

h) The `less` command will let you read text files a page at a time. You'll recognize the user interface from reading manual pages. Use `less` to scan through a few pages of a couple of your books to verify that they copied correctly and are intact[3].

Task 7. Installing software

As part of this unit, it is expected that you learn to install software packages and libraries to support your use of Linux. There are three programs that you are likely to want to use to install programs:

- Ubuntu Software Centre – this appears in the main menu and works similarly to the Microsoft, Apple, and Android app stores. Notably, unlike the other package managers listed here, it includes both free and paid-for software.
- Synaptic Package Manager – harder to use than the Software Centre, but gives you a lot more control over the installation process. Access it via **Main menu** → **System** → **Synaptic Package Manager** (you may need to install it first using `apt-get`).
- `apt-get` – command-line package manager. This needs to be run under `sudo` as installation will generally require writing to system directories. It is usually the quickest way to unstall software if you know the name of the package you want. For example:

  ```
  sudo apt-get install my-new-package
  ```

Exercises

a) Use `apt-get` to install the `jedit` package. Run `jedit` to demonstrate that it works.

b) Use Synaptic to install the chromium web browser. (If necessary, install it first using `apt-get`.) This is an open-source version of Chrome.

c) Explore the Software Centre. Use it to install the package of your choice. Can you use Software Centre to uninstall software as well?

[3] If you want to read an epub, you'll have to download software that can do that.

Task 8. Moving files between host and client

For VM users only: Most virtualization clients allow you to share directories between the host operating system (i.e. the one running the VM client) and the OS running inside the client. Sharing a directory, folder, or drive makes it very easy to copy files on and off your VM.

You can use this technique to copy your assignment and lab files to a backup location. Doing this means that you can experiment freely with the operating system – if you get it into a state where it is unusable or even won't run at all, simply download a new copy of the VM and restore your files.

Exercise

a) Locate the online documentation for your virtualization software and find out how to share directories/folders. To get you started, here are links to three of the most commonly-used VM clients:

- VirtualBox: `https://www.virtualbox.org/wiki/Documentation`
- VMware Workstation Player: `https://www.vmware.com/support/pubs/player_pubs.html`
- Parallels Desktop: `http://kb.parallels.com/au/`

b) Copy the files from Task 5 to the host computer.