✓ Done

Tutorial 0

Getting Started

Introduction

The main purpose of this practical is to introduce you to the computing environment at UNE. You will use the ideas in this practical again and again throughout the unit, so you should make every effort to understand not only *what*, but also *why* you are doing what you are doing at each step.

Each practical session includes some exercises that must be completed, and all practical from week 2 include an exercise that must be submitted for assessment (through the appropriate section of myLearn). Completing each exercise will help ensure you understand the content of this unit, and will help in preparation for your assignments and exam.

Accessing a Unix-based system

There are many options to access a Unix-based system as required for this unit. This includes installing or running your own Unix-based system (perhaps in a virtual machine), or connecting to the UNE server as described below.

Even if you have your own Unix system that you can use for this unit, it is recommended that you connect to UNE's server, as described below, because this is the environment where your assignments will be tested.

Connecting

All UNE students have access to a computer system called *turing* (not *turning* - it is named after the famous computer scientist Alan Turing). Whether on-campus or off-campus, you have access to the exact same resource. Follow the instructions below to connect to *turing*.

Armidale Students

Students have access to the computer labs (MCL1, MCL2, or MCL3) whenever other classes are not using them.

Students using the computer labs at UNE should read the following information regarding starting and ending your computer lab session.

1. The computers in the computer labs should remain on at all times.
2. We will be using Linux for all our lab sessions. From Windows, start X2Go, click on the turing session, and enter your UNE login and password (your login is what comes before @myune in your email address).
    1. If you can't see a turing session listed, follow the instructions in our Computing and Data Science Community myLearn page to create one.
3. When finished, you should log out of both the Linux session and Windows. Do not shut down the computer and/or turn off the power.

Connecting from Off-campus

You can connect to *turing* either of the following methods:

1. **Preferred** Using X2Go
2. Using ssh

Using X2Go

Detailed instructions for connecting to *turing* are available through our Computing and Data Science Community myLearn page (you may need to enrol yourself to see this content). Below is a brief summary only.

X2Go provides you with a remote desktop connection to *turing*, giving you the same access to the UNE computer science computing resources as if you were in a UNE computer lab.

Instructions for connecting to *turing* using X2Go are avaialble through the Unit Resources section of myLearn, and can be summarised as:

1. Download and install X2Go client.
2. Start X2Go and create a new session with the following information:
    - Host is *turing.une.edu.au*
    - Login is your UNE login name (no @myune.edu.au)
    - Session type must be XFCE
3. Start the session and enter your UNE password

Your UNE login name is the part of your UNE email address before the @myune.edu.au. It contains only lower case letters and numbers.

After you start X2Go, you should get a window containing a desktop on *turing*.

The first time you connect, you may be asked if you want to use the default desktop. You should say yes.

Using ssh

On UNIX-like systems (such as Linux or OSX), you can use the `ssh` command to connect to *turing*. You should use a command similar to:

```
ssh -X UNEUserName@turing.une.edu.au
```

On Windows, you can use PuTTY instead.

Getting Started With Linux

To complete this practical, you will need to work from within a terminal window. There should be an icon in the bottom panel to start a new terminal, or the option should be available either through the top menu, or by right-clicking on the Desktop.

When you see a blinking rectangle (called the *cursor*) sitting to the right of something like:

```
UNEUsername@turing %
```

or

```
UNEUsername@turing.une.edu.au $
```

you are ready to begin (note: the exact text can be customised, so may be in a different format). This text is called *system prompt*, and indicates that the computer is ready to accept your command.

When the system prompt is displayed, the system is ready to accept a command. Once you enter a command (type the command in and press Enter/Return), the system will execute the command (if it is able to) and, when finished, display the system prompt, ready for another command.

Linux Commands

This section provides an overview of some of the Linux commands that may be useful for this unit.

man

The UNIX `man` command is one of the most useful commands available on the system. It displays a manual page for commands. It is important that you learn how to use this command effectively.

Unfortunately, many of the manual pages are written by people who assume that everyone is as familiar with the system as they are. This can be frustrating and it may take some time before you become accustomed to the style used in these pages. As your understanding of UNIX increases you will find these pages more and more useful. Have a look through the different sections of the manual and try out some of the switches (e.g. `-k -f` etc).

Try reading the manual page for man itself by running the command:

```
man man
```

You can use the arrow keys to scroll through the man page, and press q to quit.

ls

The `ls` command lists all the files in your current directory. Read the man page for *ls* and the different switches available (as explained below). An important issue in a multi-user operating environment like Unix is setting and changing file access permissions. The way you set access permissions on a file will determine who can write to the file, who can read it and if it is a program or script who can execute it. You can see the permission associated with a file by executing an `ls -l` command. Executing this command in a *practicals*directory may produce:

```
total 16
-rw-r--r-- 1 cosc110 admin 2474 May 12 14:02 week2
-rw------- 1 cosc110 admin 2474 May 22 09:36 week3
```

This tells us that all of the files can be read (r) and written (w) by the owner (cosc110) and that the file "week2" can also be read by the group "admin" and everyone else. The permissions are shown in the first field, and consist of four sections:

1. File type ("-" represents a regular file, "d" would mean a directory)
2. Read, write, and execute permissions for the file owner (rwx, with a - depicting that the permission is not granted)
3. Read, write, and execute permissions for the file group (rwx, with a - depicting that the permission is not granted)
4. Read, write, and execute permissions for everyone else (rwx, with a - depicting that the permission is not granted)

For example,

```
-rwxrw-r--
||  |  |
||  |  ---> The rights of everyone (only read)
||  ------> The rights of the file's group (read and write)
|---------> The rights of the file's owner (read, write, and execute)
----------> The file type (- = regular file, d = directory, etc.)
```

Suppose there is a file called *report* in your directory with the following access permissions (if there is not, you can create one by running the command `touch report`):

```
-rw-r--r-- .......... report
```

Now, if you want to change the access permissions for all users (a) to have read and write access, you could execute:

```
chmod a+rw report
```

If you check the permissions (using *ls*), you should now see (the ...s will be replaced with actual values):

```
-rw-rw-rw- .......... report
```

Similarly, if you want to remove permissions, you can use `-`.

See what happens when you execute:

```
chmod a-rw report
```

There are a number of ways to change a file's permissions. You must be the owner of a file (or a superuser) to change the file's permissions. See the man page for the command `chmod` to find out how to change a file's access permissions. Try it out on a test file. Also, try changing the permissions of a directory and see what happens. For example, what does changing the execute permission on a directory do?

Directories

You may have guessed that there is no meaning in giving execute permission for a directory because it is not an executable program. However, the meaning for the execute permission on a directory is that you are allowed to search through the contents of a directory (so it is a sort of an access permission). For example, if you have a directory called *practical1*, you can give the following command:

```
chmod a+rwx practical1
```

to ensure that anyone can read, write, or list files in the directory *practical1*. If you do not have a directory with that name, you can create one using `mkdir practical1`.

cd

UNIX-based systems have a concept of a present working directory. Rather than having to specify the complete path of files (i.e. a path that begins with the root directory /), you can also use relative paths from the present working directory.

For example, on *turing*, the home directory for the user *cosc110* has the full path */home/unit/cosc110*. If this directory contained a file named *practical1*, you could access it with the path */home/unit/cosc110/practical1*(or *~cosc110/tutorial1*, since *~cosc110* will reference the home directory of user *cosc110* - to access your own home directory, you don't even need a username: ~ will refer to your home directory). However, if your present working directory was *~cosc110*, then you could refer to the file simply as *practical1*.

`cd` is used to change your present working directory. Explore the man page for `cd` to see how it can be used.

pwd

In order to find the absolute pathname of the present working directory, you can use the command `pwd`. It will simply list the full path of the directory you are currently in.

echo

`echo` is a useful command. It can be used to display messages on the screen and print out the value of shell variables (shell variables are described below).

Try the following:

```
echo Hello world
```

cp

`cp` copies a source to a destination. For example, if you have a file named *report*,

```
cp report report2
```

will create a new file (or overwrite an existing file) named *report2* that has the same contents as *report*.

Explore the command line options for `cp` (the `-r` switch, which performs a recursive copy, can be very useful).

mv

`mv` moves a file (or files). This works much like `cp`, only the source will not exist after the command completes. Explore some of the options listed in the `mv` man page too.

cat

`cat` concatenates the files passed to it to standard output. For example, if you have a file named *report*,

```
cat report
```

will copy the contents of the file to your screen.

less

Sometimes the contents of a file will not fit on a single screen. Rather than having it scroll past so you can't read it, you can use a pager (such as `more`, or the more modern `less`), to scroll through one page at a time.

```
less report
```

grep

`grep` is useful when you want to search through files to find a particular piece of text. For example, you can use

```
grep main *
```

to search through all files in the current directory (represented by the `*`) for any lines which have the sequence of letters "main" in them. This is a very simple search, but grep and related program (e.g. `egrep`) can use much more complex and powerful regular expressions to do a search.

head and tail

The `head` utility displays a specified number of lines from the start of a file. For example,

```
head -n 5 report
```

will display the first 5 lines from the file *report*.

Similarly, `tail` displays a number of lines from the end of a file. For example,

```
tail -n 5 report
```

will display the last 5 lines from the file *report*.

sort

The `sort` utility is used to display lines from a file in sorted order (usually alphabetic).

uniq

The `uniq` utility displays the contents of a file, skipping adjacent duplicate lines.

tar, compress, gzip, bzip2

`tar` was originally designed to create archives on tape (it stands for *tape archive*). You can tell it to create a file instead using the `f` switch. For example,

```
tar cf mybackup.tar *
```

will create (c) a file (f) called *mybackup.tar* that archives all of the files in the current directory. You can untar the file using the extract (x) switch:

```
tar xf mybackup.tar
```

Tape archives simply contain a copy of the files in the archive. This uses as much space as the original file. However, there are tools that can compress files (leaving the same information in the file, but using redundancy in the data to reduce the space required to store it). Some common compression programs are `compress` (which makes *.Z* files), `gzip` (which makes *.gz* files), and `bzip2` (which makes *.bz2* files). It is quite common to see compressed tape archives (e.g. *mybackup.tar.gz* or *mybackup.tgz*).

In fact, it's so common, `tar` has an option to compress files automatically:

```
tar czf mybackup.tgz *
```

will archive and compress all files in the current directory to a file named *mybackup.tgz*.

wc

`wc` displays the number of lines, words, and characters in a file. Read the man page to see how you can control `wc`.

Exercises

Use material from this week's lectures to perform the following exercises:

1. List the contents of the current directory
2. Change to the root directory and display its contents
3. Display the contents of the *etc* subdirectory
4. What is the absolute pathname of the file *passwd* in the *etc* subdirectory? Test your answer using `ls`.
5. From the root directory, what is the relative name of the file *passwd*?
6. Change (`cd`) to your home directory
7. What is the absolute pathname of your home directory?
8. ~ expands to your home directory. ~*username* expands to the home directory of a particular user. List (`ls`) the contents of your home directory
9. Make a subdirectory off your home directory called *cosc110prac0* and make it the present working directory
10. List the contents of your home directory from your present working directory
11. Using a Web browser on *turing*, download the slides for this week's lecture from the COSC110 myLearn page. Make sure you store the file in the *cosc110prac0* directory. Use the utility `xpdf` to view the slides (`man xpdf` could be helpful)

12. Remove the slides from the *cosc110prac0* directory using a command
13. Use a text editor (perhaps `nano`) to create a file called *UNIXCommands*. Enter the following content:

```
cd          change directories
pwd         present working directory
mv          move
cp          copy
less        list file contents
ls          list directory contents
mkdir       directory-name
rm          remove a file
```

14. Display the contents of *UNIXCommands*
15. View the man page for `wc`
16. Use `wc` to count the characters, words, and lines in *UNIXCommands*
17. Use `grep` to display just the lines of *UNIXCommands* containing the text "file"
18. Edit the file UNIXCommands and add:

```
grep        search for a pattern in a text file
wc          count lines, words and characters in a text file
```

19. Create subdirectories so your home directory includes the following:

```
         $HOME
        /    \
       /      \
      /        \
     cosc110   (other subdirectories/files)
    /       \
   /         \
  /           \
assignments practicals
```

20. Don't forget to log out (either type `exit` or press Ctrl-d (i.e. hold down control and press d)

Last modified: Thursday, 1 February 2024, 10:53 AM