

CITS4407 Open Source Tools and Scripting Introduction

Unit coordinator: Arran Stewart

Overview

Focus of the unit: this unit focuses on the philosophy, design, tools, and practices that enable and facilitate the success of open source software, which runs much of the world's computer infrastructure.

We look at topics including:

- use of the shell as a programming language
- the use of the file system and pipes to support interprocess communication
- fundamental software components
- tools supporting the software development and maintenance process
- the importance of consistent interfaces to support software integration.

Motivation

Why learn about these things?

- Allows us to automate complicated tasks that would require arduous manual labor.
- Provides a set of small commands and utilities that can be combined in unlimited ways to perform complex custom tasks.
- Extremely useful computer skill that will be relevant many years from now.
- Means we are not limited to tools provided by software suppliers - we have our own toolbox, and can create our own tools.
- Very useful in fields like data science - a major portion of work is spent using tools to obtain and process data, and convert it from one form to another.

Admin

Unit Information

Unit Coordinator: Arran Stewart

Contact: arran.stewart@uwa.edu.au

Phone: +61 8 6488 1945

Office: Rm G.08 CSSE Building

Consultation: Email for an appointment.

Unit webpage: accessible via GitHub, at

<https://github.com/cits4407>

Announcements

Announcements will be made in lectures, and on the unit help forum, [help4407](#).

It's important to check the forum regularly – at least once a week – and it is possible to set up an email subscription so you're alerted when new postings are made.

Unit contact hours – details

Lectures:

- You should attend one lecture per week – you should either attend in person, attend online (we will use Zoom), or watch the recorded lecture. (Recorded lectures are available via the university's LMS, at <https://lms.uwa.edu.au/>.)

Workshops:

- You should attend one lab/workshop each week, starting in week *two*.
If there is room available for you, you are welcome to attend other lab sessions as well.
- In the lab/workshops, we will work through practical exercises related to the unit material. If you have a laptop, it will be very useful to bring it! But you can use lab computers if not.

Non-timetabled hours

A six-point unit is deemed to be equivalent to one quarter of a full-time workload, so you are expected to commit 10–12 hours per week to the unit, averaged over the entire semester.

Outside of the contact hours (3 hours per week) for the unit, the remainder of your time should be spent reading the recommended reading, attempting exercises and working on assignment tasks.

Best sources of information

- **Unit webpage.** Your first point of call for information.
- **Ask in lectures or labs.** Often the quickest way of getting an answer!
- **Help4407.** The discussion and question forum for the unit – if you have questions not answered on the webpage, and your question would benefit other students, ask it here.
- **Unit coordinator.** If you have questions that are not appropriate for the discussion forum, the next best avenue is to email the unit coordinator (me), or arrange to come to a consultation (either in person, subject to Covid restrictions, or by telephone or online).

Textbook

There is no one textbook that covers all the content of this unit.

But an open-sourced textbook by William Shotts is helpful for many topics:

- William E. Shotts, Jr, *The Linux Command Line: A Complete Introduction*

See the website for details: <https://cits4407.github.io/#textbook>

It's available for download as a PDF, or via the UWA library, or for purchase as an e-book or paperback (either from Amazon, directly from No Starch Press, or from second-hand booksellers).

Assessment

The assessment for CITS4407 consists of an online quiz (worth 10%, done on LMS, due in week 3), two assignments (due in weeks 7 and 12), and an exam: see the Assessment page at

- <https://cits4407.github.io/assessment/>

for more details.

Assessment

In general, assessments will be due at 5pm Friday on the week they are due, but check the Help forum for more information closer to the date.

Schedule

- The current unit schedule is available on the unit website:
<https://cits4407.github.io/schedule/>
- The schedule gives recommended readings for each topic: either chapters from the Shotts textbook, or extracts or webpages. Your understanding of the lecture and workshop material will be greatly enhanced if you work through these readings prior to attending.

Programming languages

We don't assume *any* familiarity with programming, but hope that by the end of the course, you will have a familiarity with [Bash](#), written by Brian Fox and released in 1989.

If you are also doing (or have already done) Python or Java programming, then learning bash should be fairly straightforward – but we will go at a pace that everyone should be able to keep up with.

(If you want more challenging exercises, let me know!)

Why learn bash?

Bash is a very widely used *Unix shell* – a program that accepts commands written via the keyboard, and passes them to the operating system (or *OS*) to carry out.

Most operating systems (including MacOS X and Windows) do provide a shell, but non-Unix operating systems tends to place less emphasis on it.

On Windows, the default shell is called `cmd.exe`, and on MacOS X, the default shell is called [Zsh](#), and is related to Bash.

Why learn how to use a Unix shell?

- On Unix-like systems, using a *shell* is still the primary way of getting many tasks done.
- Using a shell is much more *powerful* than trying to do things purely through the GUI (Graphical User Interface), and often much faster.

Why not some other language?

- You might ask, Why not just use Python? Or Java?
- Any task you can do in Bash, it's always *possible* to do using another language – so you could use Python, or Java, or many other languages if you wished.
- Unix shells integrate very tightly with the operating system, and make it very quick and easy to control what the OS is doing.

Operating system

The operating system on which all assignments will be marked, and on which they are expected to run, is [Linux](#) – specifically, the Ubuntu 20.04 distribution of Linux.

You can use Bash and many other Unix tools from other operating systems – but it is best to make sure you do test your programs on Ubuntu Linux.

More on how to get access to it in a second.

Lab/workshops

- Much of the learning for this unit will take place in the labs, so make sure you can attend one!
- If you are attending online – the lab facilitators will be available via Zoom. (Watch the Help4407 forum for details of how to “join” the Zoom labs.)
- Some labs sessions are *purely* for online students (e.g. 2pm Friday) – others, students can attend in person, or online.

Online access to computers

- UWA IT's recommended way of accessing computers online is via **Unidesk** (<https://unidesk.uwa.edu.au>). I am still attempting to confirm that this will work to allow you access to Linux.
- However – if you have access to a Windows laptop or PC, it will likely be much quicker and easier to install Ubuntu 20.04 on your computer and use it that way.
- For each lab, there will be a Zoom meeting you can join if attending online - keep an eye on the Help forum for details.

Accessing Linux from Windows

If you have a Windows laptop or home PC, you can install a virtualised Linux operating system using what's called *the Windows Subsystem for Linux* (or WSL) – you just select a Linux *distribution* from the Microsoft Store.

More details in the first lab, next week – but for those on Windows who want to try it out before then, take a look at

<https://wiki.ubuntu.com/WSL>

and look for “Installing Ubuntu on WSL via the Microsoft Store (Recommended)”.

Access Linux from MacOS X

If you have access to a MacOS X laptop or home PC, you may want to install one of

- Parallels Desktop for the Mac –
<https://www.parallels.com/products/desktop/>
- VirtualBox – open source and free! <https://www.virtualbox.org>
- VMWare Workstation –
<https://www.vmware.com/au/products/workstation-player/workstation-player-evaluation.html>. We have student licenses available for this software.

More details in the first lab worksheet; but once one of those packages is installed, you can install a virtual Linux OS.

Access Linux from Linux

If you are already running Linux on a laptop or home PC ... then installing Linux is a done deal for you.

But if you want to ensure your assignments run correctly in an Ubuntu 20.04 environment, you might want to install Docker (<https://www.docker.com>) on your OS.

Questions?

- Take a minute to discuss with a neighbour – what would you need to know to do well in this unit? How would you define “doing well”?

Unix

Unix

Unix systems

What is Unix?

- Developed in 1969 at Bell Labs (part of AT&T, the American Telephone and Telegraph Company)
- Named for the founder of AT&T's precursor, the Bell Telephone Company, founded by Alexander Graham Bell in 1877.
- Rewritten in C in 1973 (before that, assembly language)

Unix systems

From Wikipedia:

Unix systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication (IPC) as files; and the use of a large number of software tools, small programs that can be strung together through a command line interpreter using pipes, as opposed to using a single monolithic program that includes all of the same functionality. These concepts are known as the Unix philosophy.

Unix-like

Again, from Wikipedia:

*A Unix-like (sometimes referred to as UN*X or *nix) operating system is one that behaves in a manner similar to a Unix system, while not necessarily conforming to or being certified to any version of the Single UNIX Specification.*

OS components

An operating system distribution is normally divided into (at least) two major portions:

- A *kernel* - the “master control” program, which starts and stops processes, handles low-level interaction with hardware, etc.
- *Applications* - many modular tools and programs

Each program runs within its own *process*

Programs

- Nearly every thing you use in Unix or Linux is really an external program (not a shell command or part of the kernel).
- Most of these communicate with the outside world in just four ways:
 - They get arguments on the command line
 - They receive input from standard input
 - They send output to standard output
 - (They also send error messages to standard error).
- They are small, reusable pieces that you can assemble in any way you like to do complex tasks.

Files

- Unix treats almost everything (programs, hardware devices, kernel information, directories) as a type of *file*.

Exercises

- The best way to learn how to use a Unix or Unix-like system is through practice.

Open Source

What is “open source”?

open-source əʊp(ə)n'sɔ:s / ► adjective Computing
denoting software for which the original source code is made freely available and may be redistributed and modified.

— Oxford English Dictionary

What is “open source”?

There is actually some contention over what exactly “open source” means.

But the term was originally used in relation to software, and in contrast to software which is simply (monetarily) “free”.

Lots of software is *free* in the sense that it doesn't cost you any money to use; but only some of that software gives you free access to the *source code* for the software, and allows you to change it, and redistribute your changes.

What is “open source”?

“Open source” originally applied to software, but now is also used for other types of content – for instance, art works, written works other than software, and even music scores and performances.

Why use (or create) open source software

Some key ones are:

- flexibility,
- cost, and
- transparency.

Some people also assert that open source software

- is of higher quality
- has better security, and
- is more stable

than closed-source software, although this is difficult to verify.

We will look more at the philosophy behind open source software later.