What Does This Course Offer?

- How to be a Unix user—in fact *power*user: advanced commands, scripting.

What Does This Course Offer?

- How to be a Unix user—in fact *power*user: advanced commands, scripting.
- How to be a Unix programmer, using C to call up system services (system calls), e.g., I/O, file operations, launch programs, talk to another program, talk to another program on another computer. . .

What Does This Course Offer?

- How to be a Unix user—in fact *power*user: advanced commands, scripting.
- How to be a Unix programmer, using C to call up system services (system calls), e.g., I/O, file operations, launch programs, talk to another program, talk to another program on another computer...
- Gateway ~~drug~~ course to C69 (operating systems), D58 (computer networks),...

What Does This Course Offer?

- How to be a Unix user—in fact *power*user: advanced commands, scripting.
- How to be a Unix programmer, using C to call up system services (system calls), e.g., I/O, file operations, launch programs, talk to another program, talk to another program on another computer...
- Gateway ~~drug~~ course to C69 (operating systems), D58 (computer networks),...
- This course uses Linux, a Unix-like OS (though not a descendent). There are some differences. There are also many Unix descendents and other Unix-like OSes.

  ("Descendent" means source code traceable to Unix. The others started independently but implement almost the same features.)

Why Study Unix?

Unix has a yesteryear design but:

- Contains many ideas and techniques worth learning.

Why Study Unix?

Unix has a yesteryear design but:

- Contains many ideas and techniques worth learning.
- Home of many tools useful for all programmers.
  We will look at some.
  (Unix was made to support programmers.)

Why Study Unix?

Unix has a yesteryear design but:

- Contains many ideas and techniques worth learning.
- Home of many tools useful for all programmers.
  We will look at some.
  (Unix was made to support programmers.)
- Unix-like OSes ubiquitous though you don't see them:
  Most network equipments (home and industry), Kindles, Android, iOS, PS4, Steam Console.
  Favourite choice on Raspberry Pi, servers (those in data centres, "cloud").

Unix {Design, Philosophy, Style, Spirit}

Video clip: Kernighan's explanation.

Small, focused, I/O programs.

Combine them in a shell command/script for complex tasks.

To support that technique, OS fakes a file-like interface for all I/O sources and destinations, including peripherals and pipelining.

E.g., program reads input the same way, be it from files or terminals or other programs. Terminals even has fake filenames.

(Unfortunately) Terse culture, e.g., copy files with `cp`, delete files with `rm`.

Incompleteness Guarantee

This course cannot possibly go over all utilities, all system calls, all special files, all scripting tricks, all C techniques. . .

This course will only provide orientation and cover selected topics.

From them, you must learn the underlying skill (system-level thinking) so you are ready to pick up the rest upon demand.

Assignments, labs, term test, and exam can require you to learn uncovered topics (but short) on the spot and apply them.

Survival Tips

Ctrl-C can abort most programs.

Many programs support `--help` as a command line argument, e.g., try `uniq --help`

Many programs (also C library functions and system calls) have detailed doc via the `man` program (short for “manual”), e.g., try `man sort`

Those “man pages” have professionals in mind. It is normal if you can’t understand at the beginning. But learn to pick out the parts you need as you go.

Seek help early, seek help often. Think up and perform experiments, build toys to test ideas. Treat this as a *tinkering* course (esp. not just “knowledge transfer”).

Hope you will get the hang of it!