A shorter simpler course review

CS2503
Operating Systems I

VISITING STUDENTS

This module is often
NOT accepted
as a substitute
for a basic OS theory
course back home,
Just because

IT ISN'T!

Basic content

- Filesystems – where files are organised (!?)
  - Files – where you store data/info… (dis-?)organisation
  - In Unix/Linux … many things are treated as file, including devices
- Text processing to
  - Find & refine text both filenames and file contents
  - Done by Regular Expressions – formalism to specify text patterns
- Standard tools that save reinventing the wheel before scripting in the shell
  - Sed – fast non-interactive programmable editor (for full filesystem)
  - Awk – powerful programming tool, for numbers as well as text
- Scripting in the shell … here in bash, more modern, but not as portable as sh
  - Can use all of above, plus more to do almost anything you want in a system

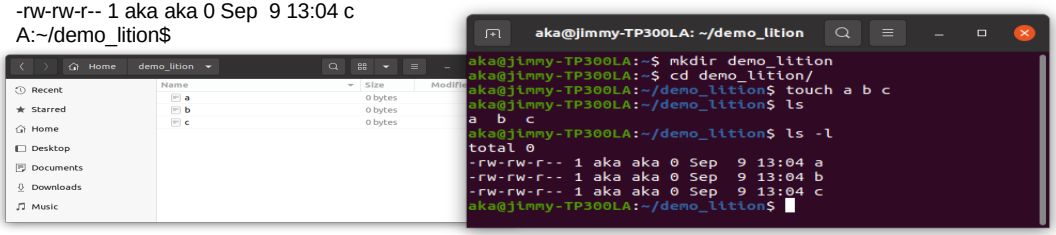
Files & Filesystems

- Filesystems – where files are organised (!?) - like a filing cabinet
  - Birds of a feather filed together – before fast find tools!
  - Even if you don't, systems configurations do with expectations
  - Files – where you store data/info… as a byte-stream/sequence
    - Can be anything, including audio/visual multimedia
  - Files can be copied, moved & removed – often accidentally
    - Move on onto another, the original is wiped,
      - Unless interactive checks enforced, (are you sure)
      - CLI interactive default is omission
      - why? For noninteractive scripts,
        - But checks should be programmed to avoid loss and chaos

Simple sample session - commands vs menu driven!

A:~$ is the prompt indicating the terminal shell is ready and waiting for input
A being the machine name, : a separator, ~ the tilde indicating home dir & $ the prompt!

```
A:~$ mkdir demo_lition                  make directory demo-lition, as we'll scrap it
A:~$ cd demo_lition/                    and cd change directory into it
A:~/demo_lition$ touch a b c            then touch up (!?) 3 files a, b and c
A:~/demo_lition$ ls                     list them  ... tab separated
a  b  c
A:~/demo_lition$ ls -l       creation time          list ... -long .. more info
total 0
-rw-rw-r-- 1 aka aka 0 Sep  9 13:04 a   shows read/write optios for user groups,
-rw-rw-r-- 1 aka aka 0 Sep  9 13:04 b
-rw-rw-r-- 1 aka aka 0 Sep  9 13:04 c
A:~/demo_lition$
```



Links – brinkmanship

- To shrink space
- But could be on the blink if original moved
  - Lost and gone forever
  - Broken links
  - Filesystem corruption
  - (not at physical block level… but logical name level)

Now where is that? Finding files

- either by name (find) or content (grep)
- Text/data – just sequence of symbols
- So advanced text processing tools
  - Optimised over the years for old systems and relatively large files…
    - Including streaming through memory hierarchy etc
      - Faster, costlier to make and run, so small, & levels…
      - Principle of locality of reference – mostly local in time and space.
  - Still applicable (even in gene research)
    - As the files are still large relative to machine memory

Regular Expressions – finding patterns

- a simple specification for text patterns
  - except there are many variations,
  - not just one for each language, but one for each tool, with two main notional standards:-
    - Basic and Extended RegEx.. BRE/ERE
    - Which of course use confusingly opposite rules at times!!
- Irregular expressions might be a better name…
  - But they are still very useful!
    - If you can get them right!

Editors… often the point in finding...

- Interactive… you must be there to control it
  - Old ones seem archaic but are efficient in odd ways
    - Vi, vim, emacs etc… more later…
      - Easy on machine, and on your body to avoid RSI
      - Ed can handle large files easily, as only editing context in RAM
- Non-interactive – program should control it
  - Sed (stream editor) based on ed
  - Can edit the whole filesystem in a flash
    - With a one (or so) liner

Other tools – used in scripting

- to avoid reinventing the wheel!!
- Awk family (name variations)
  - Simple, powerful, flexible and handy
  - Rapid prototyping
  - Tokenise data, can handle numbers as easy as text
- Many others
  - Exist – e.g. PERL – but tricky and timeconsuming
  - Emerge – but may not be used much
    - Especially in legacy code

Shell scripting – using all of the above

- But many shells,
  - Bash more modern, easier, more facilities
    - But not so portable or POSIX compliant
  - Sh – older, fewer facilities, may be easier
    - but more portable
- some variations from Linux to Unix… (Mac uses Unix)
  - e.g. when using the bash shell!!

End goal of course

- Should be able to
  - write simple scripts
  - Understand complex ones
    - And modify them to suit needs
- After that you could do almost anything …
  - if you could be bothered working it out, or searching
- The course is
  - Using all the basic construction tools
  - without really building anything complex...

But the world is a much bigger place

- Lots of:-
  - Distributions
    - https://en.wikipedia.org/wiki/List_of_Linux_distributions#/media/File:Linux_Distribution_Timeline.svg
  - GUI's & window managers
  - Shells
- Not to mention options
  - Configurations – can configure almost everything
- And software installation tools
- Would like to do much more,
  - but do enough so you can do the rest!

- Mostly 2 letter cryptic commands
  - With lots of unintelligible flags
  - And a minimal help system
- But can jam them together in a long one-liner
  - Piping output from one into another
  - To do a lot interactively at the screen
  - & if takes too long, run in the background
    - (or open a new terminal session)
- Or write a script, and but you and it can run away!!!