
LINUX Advanced Topics

Bruce M Gittings

LINUX Advanced Topics

- This section of the Introductory Computing Course will look at:
 - Scratch space
 - Controlling LINUX Processes
 - Joining commands together
 - Redirecting output to a file
 - Automating processes with a shell script
 - The need to edit files
 - Editing on LINUX

Scratch Space

- Scratch space provides large amount of data storage for short-ish periods
- Just as you have space here:
 /home/yourusername
- So you also have space here:
 /scratch/yourusername
- There is no backup of this data, but it probably won't be deleted
- You can access 'scratch' from your M: drive via a 'shortcut' folder or connect it as a virtual PC drive:
 \\s1xxxxx.scratch.geos.ed.ac.uk\scratch\s1xxxxx

LINUX Processes

Terminology:

- All applications in LINUX start one or more **processes**
- This happens in Windows too, but in LINUX it's easier to see how they work and you have far more control over the processes
- A knowledge of processes gives **you** flexibility of operation
- It also lets **you** keep the LINUX machines running smoothly and **fast**
- A **background process** is one which runs behind the scenes putting its output in a file having been spun off by the user
- A **batch process** also puts its output in a file, but takes its input from another file as a series of commands)

Controlling Processes

- Can start, stop, pause, change priority of all the processes that belong to you
- Why bother?
 - Because sometimes it's very useful to hide your model in the background (still working) while you leave the building.
 - Or have a web browser open on the relevant web help page as you try and understand the LINUX command you just learnt
 - In other words, **you** can control which of **your** processes are given priority (run faster)
 - Or kill off processes which have crashed or run amuck
- Allows more efficient multitasking - you can set the priorities of processes with the `renice` command so that unimportant tasks do not hog the servers processor power

Process-related Commands

- The **ps** command allows processes to be examined

ps -af

allows you to find out exactly what other people are up to!

- The **kill** command is vital for tidying up when things go wrong

kill -9 5226

unambiguously kills the process with PID=5226

- The PID (Process Id) is displayed using **ps**
- You can only kill your own processes !

Controlling Batch Processes

- Batch processes run single commands, sets of commands (scripts) or applications
- They cannot (easily) run applications which require graphical input
- Input is taken from a file as a series of commands
- Any output, which would ordinarily go to the screen, is written to a file
- They can be started automatically at any pre-defined time
- The `at` (simpler) and `crontab` (more complex) commands allows batch processes to be defined on LINUX
- `atq` shows what is queued, `atrm` removes a job and `kill` kills it if already running

Joining LINUX commands together

- Terminology: a **pipe** is a way of taking the **output** from one command and feeding it directly as **input** into another command
- A pipe is represented by the vertical bar on the bottom left of the keyboard: |
- For example:

```
who | grep '^s'
```

shows only the users beginning with the letter "s"

Saving output to a file

- Almost all the commands you used this morning sent their output to the screen
- Extending the concept of a pipe, you can easily redirect output of a command to a text file for future use
- For example,

```
grep -n 'Bruce' index.html > file.txt
```

Would output all lines from my web page containing the word 'Bruce'

- ***Pipes*** and ***redirection*** used together can give you powerful search and analysis tools for only a tiny bit of typing...

Shells and Shell Scripts

- LINUX commands are typed at a shell prompt
eg. `bash$`
- The shell passes the command to the LINUX kernel for execution
- Different shells are available, offering different facilities eg.
 - customising the environment
 - define command aliases
 - edit and reissue previous commands
 - automatically complete the command line
- It is also possible to run a sequence of commands together into a ***shell script***

Shell Scripts

- This is a very simple bash shell script:

```
#!/bin/bash
echo "hello, $USER. I wish to list some of your files"
echo "Listing files in the current directory, $PWD"
ls # list files
```

- This example is more complex:

```
#!/bin/bash
OPTIONS="Hello Quit"
select opt in $OPTIONS; do
    if [ "$opt" = "Quit" ]; then
        echo done
        exit
    elif [ "$opt" = "Hello" ]; then
        echo Hello World
    else
        clear
        echo bad option
    fi
done
```

Modelling - Visualisation

- Several programming environments are available for scientific data visualisation and modelling:
 - Python (free and open-source)
 - MATLAB (licenced)
 - IDL (limited licenced; GDL free and open-source)
 - Java (free and open-source)
- These usually run on LINUX
- Research groups favour different environments based on existing code base and models
- Python is extensible, with MATPLOTLIB resembles MATLIB, and is used to customise tools like ArcGIS

Using an Editor

- Editing involves modifying programs or data, as against word-processing which involves creating documents
- You may also need to prepare or modify data files as input to a program which undertakes analysis (eg. statistics package, GIS etc.)
- You don't want to use a word-processor because it will introduce *formatting characters*
- You could use a spreadsheet, but this introduces structure to your data
- An editor lets you get at the raw data

ASCII versus Binary files

- Generally:
 - ASCII files (or text files) are those which contain data you can understand
 - Binary files contain data which the computer understands
- Data files may be ASCII or binary, as can programs
- Binary files compress information into a more compact form
- You shouldn't try to directly print binary files
- Binary files are usually very specific to the software which created them (eg. Photoshop)
- Word-processing files are actually binary, because they contain hidden formatting information

Editing files

- PSPad is a powerful PC editor
- There is also another useful editor called TextPad (and WordPad and NotePad)
- PSPad does many things familiar from a word-processor, but also has advanced features such as search-and-replace across multiple files
- Slightly unusually, PSPad will allow you to edit binary files, but this is a dangerous operation

PSPad – Useful Facilities

- Opening multiple files, allocating them to tabs and reopening them when PSPad is next used
- Showing line numbers
- Wrapping / unwrapping of long lines in data and program files
- Find and replace over multiple files
- Differences between files
- Colour coding of program files
- PSPad can be run directly from your USB stick without installing on a machine

Text Editing on LINUX

- Preparing shell scripts, programming, configuring the system etc. all need text files to be edited
- Could use PSPad from a managed desktop PC to edit files on your network drive assuming this is attached to LINUX
- This is an indirect way of editing files
- LINUX and PC text files differ in the way they handle returns/end of line characters
- Thus you can easily get confused and wonder why things aren't working!
- Windows based PSPad can read and save files in either PC or UNIX format, take care which!

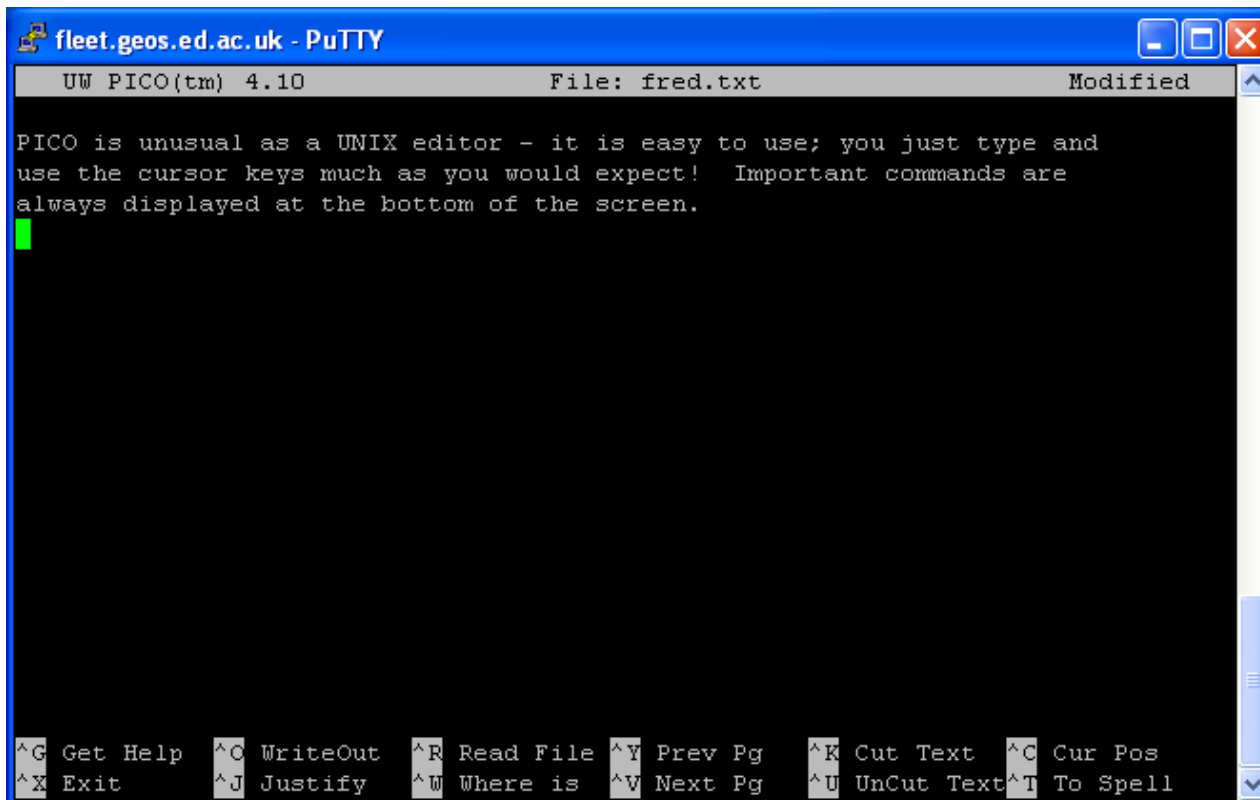
LINUX Editors

- There are many editors which run directly on LINUX
- Like cars some people get excitedly evangelical about their favourite editor and despise others!
- Some good choices:
 - pico (easy-to-use, widely available)
 - emacs (long established, v. sophisticated, but more difficult to use. Supports X-Windows)
- Don't seem as friendly - use keyboard commands, but once you know these it is extremely fast to use
- Quicker than PSPad – less clicking – but only when you know what you're doing!
- Emacs is also available for the PC if you get to really love it

Pico

- A screen-based text editor
- Available on most LINUX systems
- Unusual for a LINUX editor it is easy to use!
- The default editor for the PINE mail system
- Written for this purpose by the University of Washington

More Pico



```
fleet.geos.ed.ac.uk - PuTTY
UW PICO(tm) 4.10      File: fred.txt      Modified

PICO is unusual as a UNIX editor - it is easy to use; you just type and
use the cursor keys much as you would expect!  Important commands are
always displayed at the bottom of the screen.

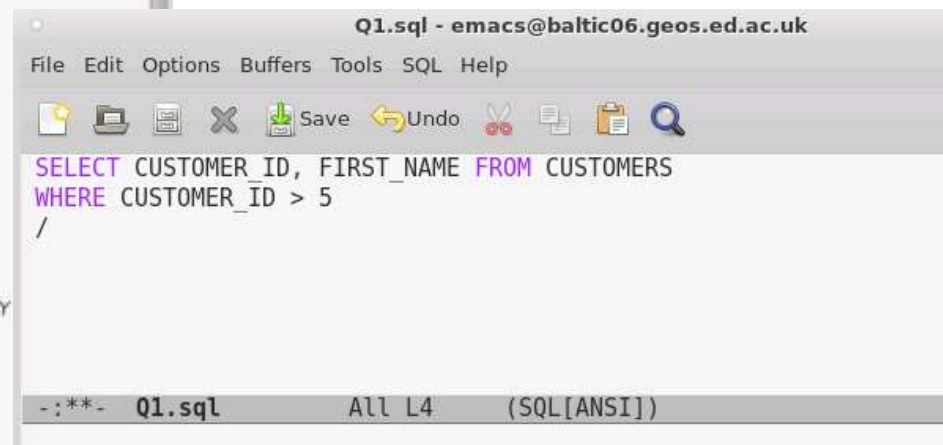
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Pg  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where is   ^V Next Pg  ^U UnCut Text ^T To Spell
```

- Possible commands are displayed at the bottom of every screen
- Straightforward help info is easily accessible

Emacs

- There are graphical and command-based versions
- Developed in 1976 as a set of macros for the DEC's 1962 TECO editor
- Can be more difficult to use, but lots of facilities
- Over 10,000 built-in commands
- We will use this in the core GIS courses *Spatial Modelling and Analysis* (for writing database queries) and *Technological Infrastructures for GIS* (for python programming)
- In remote desktop / xrdp, give the command **emacs**

More emacs



- Commands are on menus
- Features syntax-highlighting – very useful for programming
- Tutorials and manuals are available