GNU history

- 1970s software copyright legal changes
- 1980s Unix licensing restrictions
- 1985 FSF formed to develop a permissively licensed Unix
- 1985 Emacs
- 1987 GCC
- 1989 GPL
- 1989 Bash
- 1991 Mass internet adoption (magnified permissive advantages, sped up dev)
- 1992 coreutils
- 1992 Linux (separate to GNU but used GPL)

Key concepts above are permissive licensing and the internet.
These are quite symbiotic and thus a core strength and use case of Linux is the internet network stack.

Consequently Linux is ubiquitous, and its use permeates the internet and most companies, so familiarity with the interface is a really useful skill.

Unix Model

- any computer is just logic manipulating data
  - Whether it’s hardware or software
  - programming paradigms (functional, procedural, OOP, ...) just abstractions of that
- Unix abstraction is a little higher level
  - processes and files map to logic and data
  - more accurately processes and file descriptors
- shell is a DSL for managing processes and files
  - < and > read and write the file system namespace
    - I.e. create a descriptor from a file name and connect to a process
  - shell designed for interactive use so reads/writes tty implicitly
- Processes managed with a largely functional programming model
  - shell supports functional composition and lazy evaluation through pipes
  - consider this example: seq inf | head -n10 | tac
  - lazy evaluation supported through key SIGPIPE concept
    - https://www.pixelbeat.org/programming/sigpipe_handling.html

GNU coreutils

- The GNU coreutils can be seen as basic function set of Unix
  - Maintenance and augmentation need to carefully consider the Unix model
- Currently 108 utilities, here compared with similar commands on other systems
  - https://www.pixelbeat.org/docs/unix_commands/
- Changes over time to enhance and adjust to changing systems
  - https://github.com/coreutils/coreutils/blob/master/NEWS

Tips for handling the long and varied Unix lineage

- Long history, so benefits from config appropriate to modern systems
  - dialup, dumb terminals, small RAM no longer the case
  - tools should save lots of history
    - See https://www.pixelbeat.org/settings for bash, gdb, screen
  - powerful when combined with history navigation shortcuts
    - Settings shared between gdb/python/shell
    - See https://www.pixelbeat.org/.inputrc
  - Use colors to enhance interactions
    - At least 256 colors generally available
    - Can enable default coloring in vim/less/grep/diff/gcc/...
    - https://www.pixelbeat.org/docs/terminal_colours/

- Disparate history, so benefits from config to have consistent interface
  - Vim vs Emacs has percolated to many tools
    - Bash defaults to emacs key bindings
    - Less defaults to vim key bindings
    - Best to use one consistently and get to know well
      - Again many interactive tools (including bash) share .inputrc settings
  - Important for efficiency and RSI avoidance

Quick Demo

```
~$ # Put full exit status in prompt
~$ PS1="[\${PIPESTATUS[@]}] \w$ "
[0] ~$

~$ # Add some color to highlight issues (can set this in .bashrc)
~$ PS1="[\[\033[1;31m\]\${PIPESTATUS[@]/#0/\[\033[0m\]"\
"\[\033[1;32m\]0\[\033[1;31m\]}\[\033[0m\]] \w$ "

[0] ~$
[0] ~$ true | false | true
[0 1 0] ~$

[0] ~$ seq inf | head -n10 | paste -d, -s
1,2,3,4,5,6,7,8,9,10
[141 0 0] ~$
[0] ~$ # 141 due to SIGPIPE (13 + 128)
```