Unix
Philosophy and Workflow

Saurabh Kumar

2013-03-10

40+ years of Unix

- Unix was written in 1969.
- A unique culture and philosophy has evolved around it.
- Small, correct programs that do one thing well.
- Programs should work together.
- Programs should handle text, because text is a universal interface.

Simple != Easy

A learning curve is involved.

RTFM

- Man pages are your friend.
- Try it out. `man cat`, `man less`, `man tar`
- Man pages are divided into sections.
- You can search for man pages using apropos.
- `apropos strlen`, `man 3 strlen`

CLI

- A command line interface is very powerful.
- `ls, mkdir foo, cd foo, touch file, cp file .., mv ../file .`
- `echo ``Hello'' >> file, cat file`
- ```
  for i in `seq 1 100`;
  do
  echo "Hello"$i >>file
  done
  ```
- `cat file, cat file | less`
  ```

Concepts

- (Nearly)Every thing is a file.
- Your harddisk is `/dev/sda-z`
- Standard input, output are `/dev/stdin`, `/dev/stdout`
- Piping and redirection. `|`, `<`, `>`, `>>`
- `cat /dev/stdin`, `cat /dev/stdin >> file`, `cat /dev/stdin > /dev/stdout`
- Burning a usb image: `cat debian.iso > /dev/sdb && sync`

Other examples

- Download input file from `http://sprunge.us/cdGI` or from `https://raw.github.com/sa1/bisect-demo/master/3lpigs`
- `tr ' ' '\n' < 3lpigs | sort`
  `tr ' ' '\n' < 3lpigs | sort | uniq -c | sort -n`
- `grep 'wolf' 3lpigs`
- `sed -i 's/wolf/wolves/g' 3lpigs`
- Wildcards: `ls fi*`, `ls *.*`

Automate the boring parts

- Script what needs to be done.
- ```
  #! /usr/bin/bash
  echo ``Running task'' >> file
  ```
- ```
  at 0430
  > notify-send ``Executing task''
  > ./script.sh
  ```
- Use `cron` for scheduling periodic tasks.
- Startup scripts - `systemd`, `/etc/profile.d/`, `~/.xinitrc`

Developer workflow

Unix as your IDE

- A single large IDE vs a large number of tools that work together.
- Edit - `vim, emacs`
- Build - `make`
- Debug - `gdb, valgrind, strace, lsof`
- Version Control - `git, svn, diff, patch`

Editors

- Vim and Emacs
- Syntax highlighting, modes, macros
- Extensibility, plugins - lint, syntax checkers.
- Communicating with external programs.

Make

```
CC = gcc
OBJECTS = main.o example.o library.o
BINARY = example

all: example

example: $(OBJECTS)
    $(CC) $(OBJECTS) -o $(BINARY)

docs: docs.html

docs.html: docs.md
    markdown docs.md > docs.html

clean:
    rm -f $(BINARY) $(OBJECTS)
```

Testing

- Use scripts to set up the environment.
- Scripts to test a system.
- Testsuites
- CI- Continuous Integration
- Build, Compile, Test, Fix, Commit.

Version Control

- file1.c, file1final.c, file1_realfinal.c
- `git init`, `echo ``Hello'' >> file`, `git add file`, `git commit`
- `echo ``Hello 2'' >> file`, `git status`, `git diff`, `git add file`, `git commit`
- `git log`
- Branches, diffs, patches

Git Bisect

- Using binary search to find buggy commits.
- `git clone https://github.com/sa1/bisect-demo.git`
- `git log`, `git bisect start`
- `git bisect bad` to mark bad commits
- `git bisect good` to mark good commits.
- `git bisect reset` to reset.

Thank you

- Feedback
- http://klug.github.com