$ echo “Hello Penguicon!”

Beginner's Baptism on the UNIX Command Line

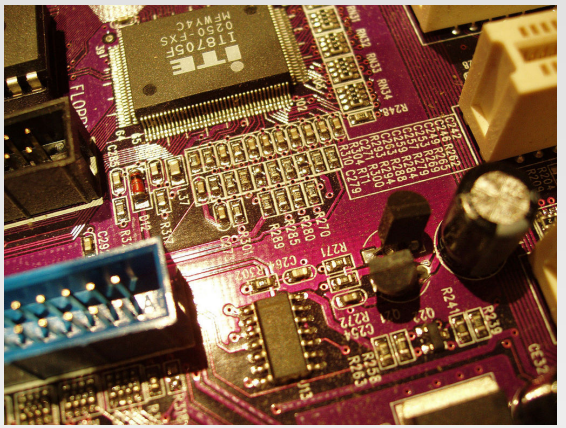
(or: how to put 30+ years of development to work for you)

Craig Maloney

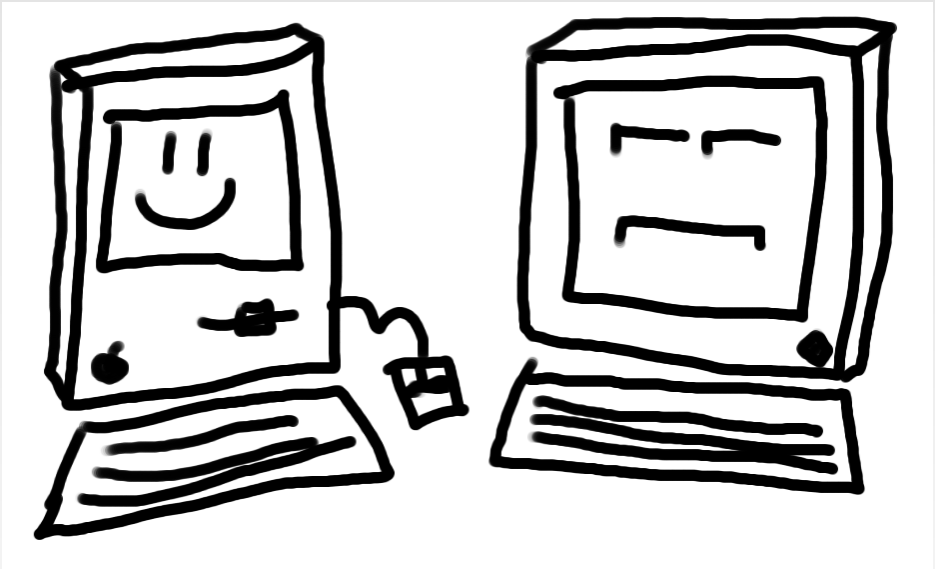
Presented at Penguicon 6.0

$ echo “What we'll cover”

- Why the command line?
- History of UNIX (the 5¢ tour)
- What UNIX is, what UNIX is not
- Shells and important concepts
- 13 important UNIX Commands
- More commands, job control, and Root access
- Users and Permissions
- Pipes
- Productivity enhancements



$ echo “Why the command line?”



- Replicable
  - Easy to script / duplicate
- Powerful
  - Can perform same action across multiple items
- Easier
  - GUI tools sometimes use command line
- Zen
  - Get into the zone

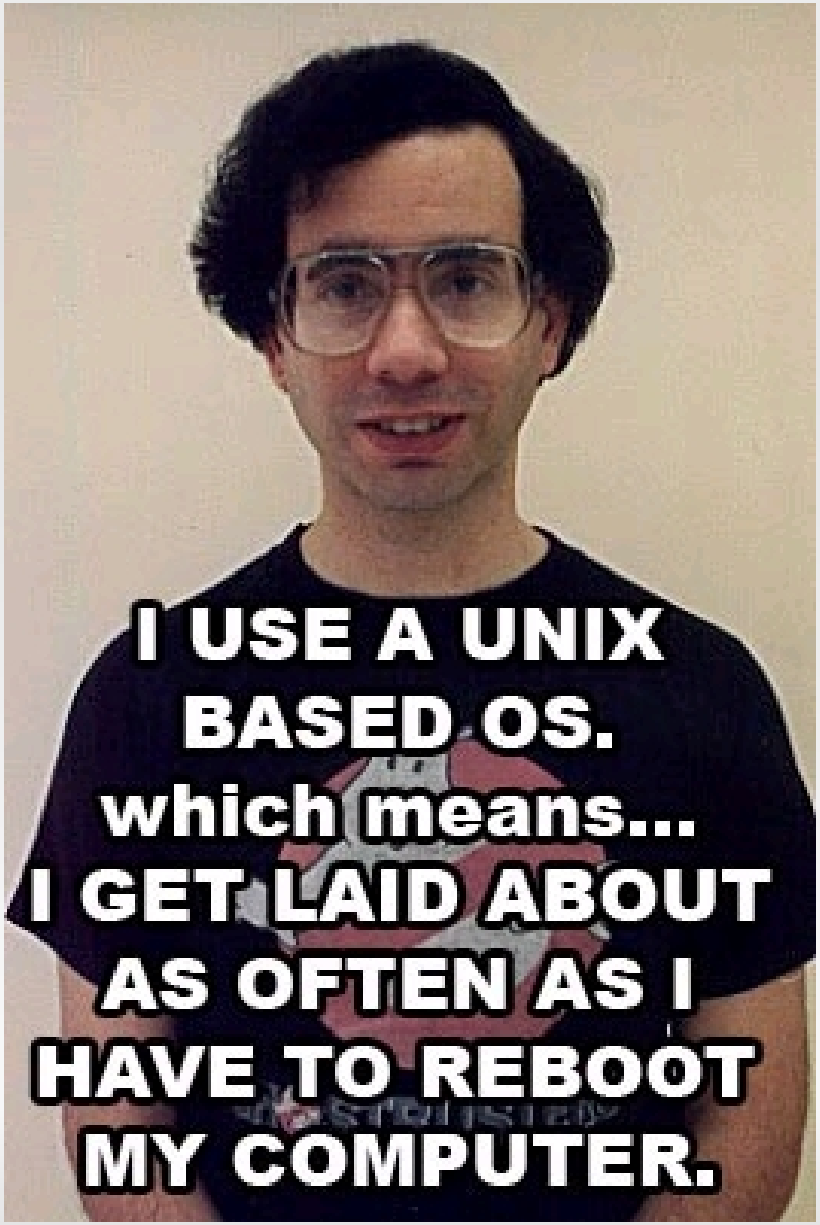
$ echo “UNIX History”

- Written by AT&T employees, including Ken Thompson, Dennis Ritchie, and Doug McIlRoy in the late 1960's.
- Rewritten in C in 1973
  - made UNIX portable to other hardware.

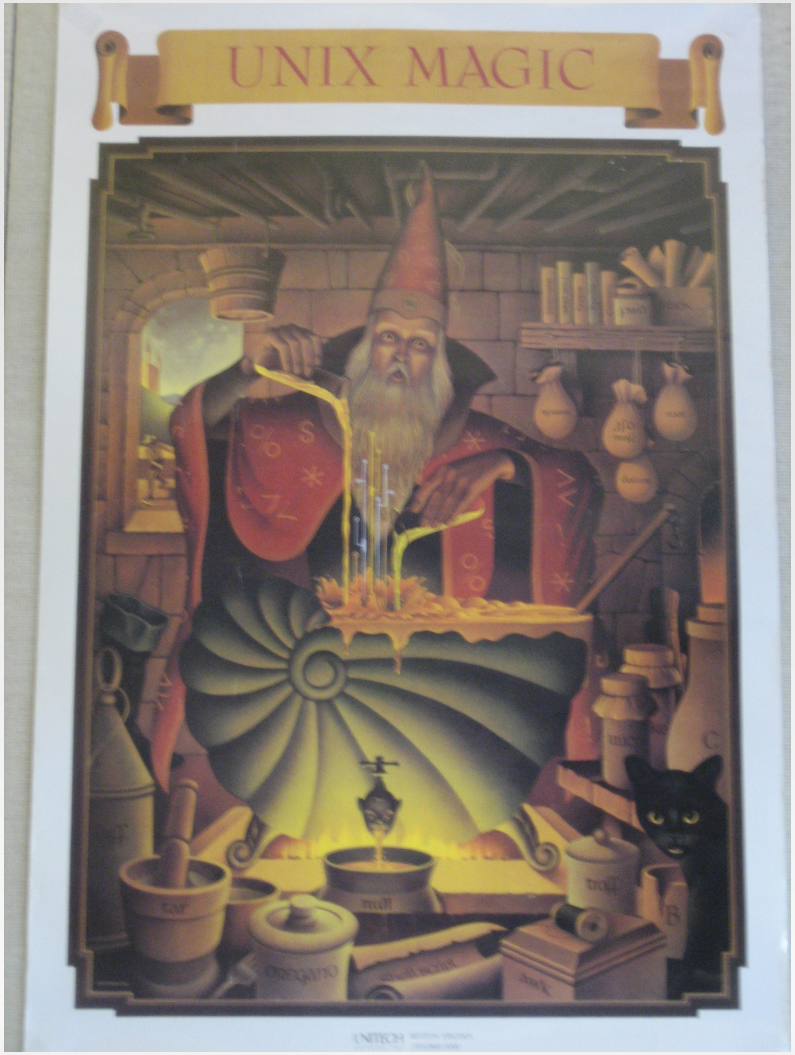
$ echo “UNIX is...”

- UNIX is...
  - Multiuser
  - Robust
  - Secure
  - Command-line driven
  - Flexible
  - EVERYWHERE!

$ echo “UNIX is not...”



$ echo “UNIX Philosophy”



“This is the Unix philosophy:

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface."

--Doug McIlroy

$ echo “Where do we start?

- Accessing a UNIX machine requires:
  - Physical or network access
  - User Account
  - A Shell (a what?)
  - A terminal program if remote (telnet or ssh)
    - Under Windows I recommend using PuTTY
    - Linux comes with `ssh`

$ echo “What's a shell?”

- A shell is the command line interface.
- Many shells to choose from:
  - We'll use `bash`.
- Many differences between shells. Won't go into them here.

$ echo “Things to keep in mind”

- Delimiters:
  - Spaces separate file names
    - `rm file1 file2`
    - use “ “ or escape (\) characters for filenames with spaces
      - `rm spacely\ file`
      - `rm “spacely file”`
  - / - Separates directories
    - `/root/dir1/dir2`
- `cASe MaTTeRS!`

$ echo “13 important UNIX commands you should know.”

- ls
- cp
- mv
- rm
- mkdir
- rmdir
- cd
- cat
- more / less
- grep
- ps
- kill
- man
- vi

$ echo “Listing Files”

- `ls` – LiSt files
  - Similar to DIR in DOS
  - Displays what files are in the current directory
- Common switches
  - `-a` lists hidden files
  - `-l` long format
  - `-al` (switches can be combined together)
  - `-r` shows files from earliest to latest

$ echo “Copying Files”

- `cp` - CoPy files
  - No indication of success. In UNIX, silence is success.
  - Can be used with wildcards, as long as the destination is a directory.
  - Overwrites destination files by default.

$ echo “Copying Files (cont.)”

- Useful switches for cp
  - `cp -r`: recursive copy
    - useful for directories
  - `cp -i`: interactive
    - Asks before overwriting files
      - (VERY useful to help prevent occasional screw-ups).

$ echo “Moving Files”

- `mv` – MoVe files
- Used to rename files.
  - `mv file1 file2`
- Also used to move files to another location.
  - `mv file1 dir2/`
- Useful switches:
  - `mv -i`: ask before overwriting files.

$ echo “Deleting Files”

- `rm` – ReMove Files
- Use with extreme caution: no recovery!
- Useful switches:
  - `rm -i`: ask before deleting.
  - `rm -r`: recursive.
  - `rm -f`: forcefully (no questions asked).
  - `rm -rf`: bye bye files.

$ echo “Making, Removing, and Navigating Directories”

- `mkdir` - MaKe DIRectory
- `rmdir` - ReMove DIRectory
- `cd` – Change Directory
- (Pretty simple, no?)

$ echo “More about Directories”

- All directories under UNIX descend from / (pronounced 'root')
- Common directories
  - `/etc` – configurations.
  - `/usr` – executables, libraries, include files.
  - `/home` – User files & data.
  - `/var` – Temp. and changing files (data).

$ echo “Displaying files”

- `cat` – conCATenate files.
- Allows multiple files to be displayed
  - `cat file1 file2 file3`.
- Much more useful later on.

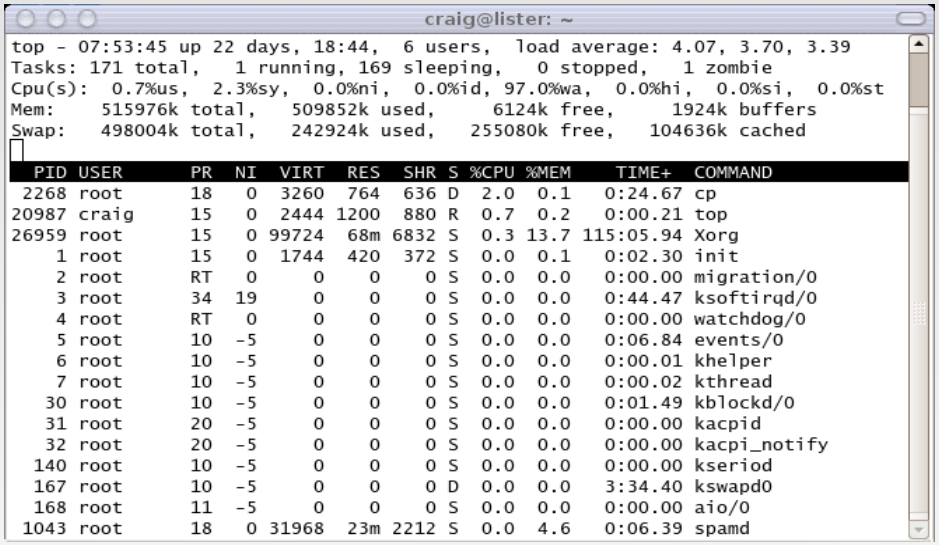
$ echo “Displaying files (cont.)”

- `more / less`
- Allows files to be displayed on the screen without scrolling off.
- `less` came after `more`, and under Linux is generally the same program.

$ echo “Searching for text”

- grep – g/re/p
- Global / Regular Expression / Print
- `grep` *text filename*
  - prints all lines with 'text' in the file filename.
- More later...

$ echo “Managing Processes”



- ps – ProceSs list
- Every task is a process.
- Every process has a number (process ID).
- `ps`: shows processes under your shell
- `ps -ef`: shows processes for all users.

$ echo “Terminate a process”

- kill – KILLs a process
- Use the process ID to identify the process
  - `kill 2112`: Kills process 2112.
- (Actually sends a signal SIGTERM to the process. It's up to the process to end.
- `kill -9`
  - Program can't intercept.
  - “Kill it alot”.

$ echo “Online Documentation”

- man – MANual
- Documentation for most all commands and their switches
  - `man ls` - Useful!
  - `man -k`: Keyword search
  - `man 5 crontab`: (File format for crontab, not the command 'crontab')
  - `man man` – it works. :)

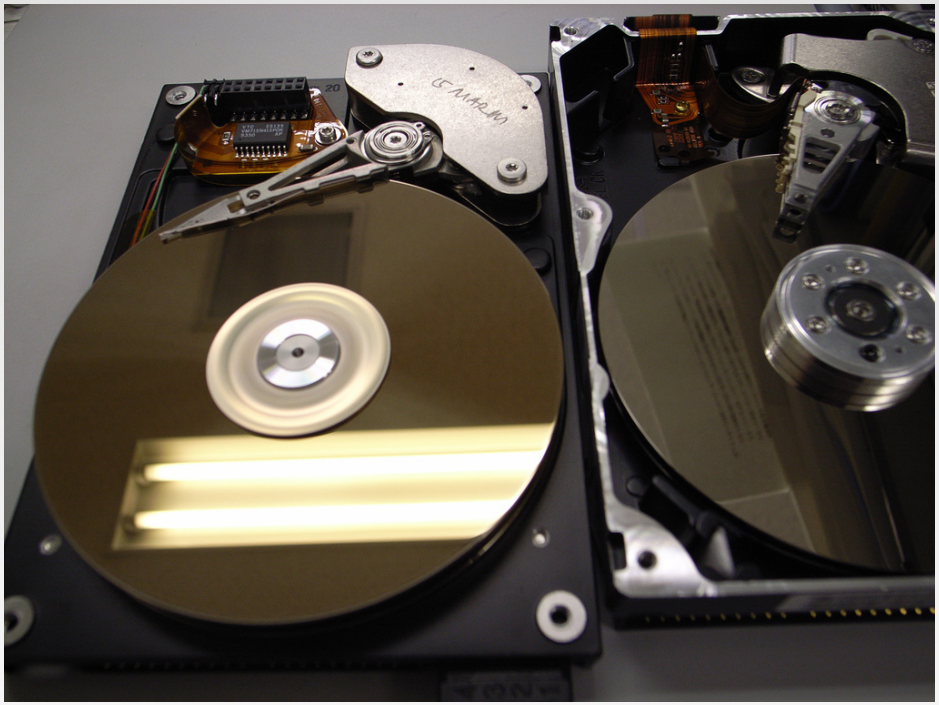
$ echo “vi – the ubiquitous editor of infamy”

- vi – VIsual text editor
- Why learn it?
  - You'll eventually get dumped by some program into vi.
- Ubiquitous
  - Shows up on every UNIX system out there.
- Infamous
  - Legendary difficulty.

$ echo “vi (cont.)”

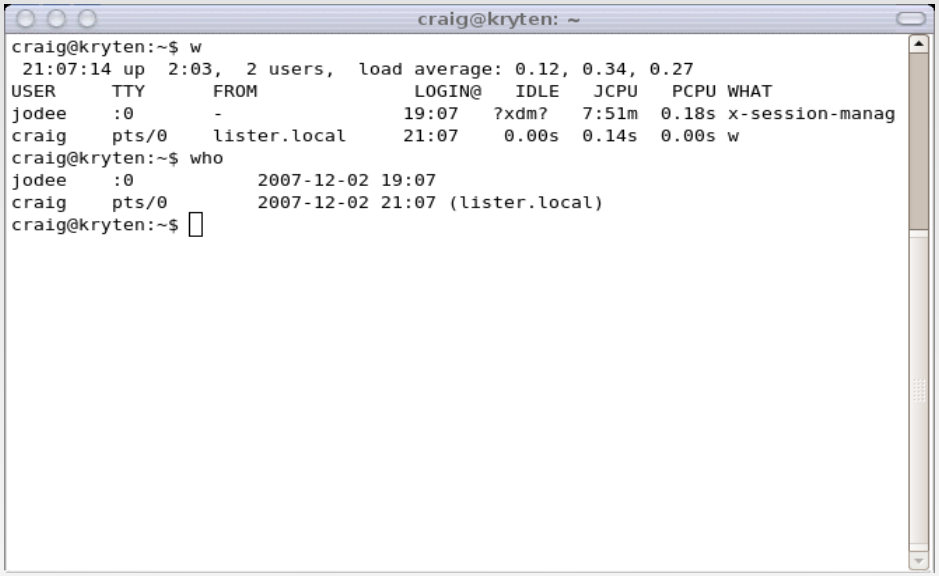
- Important key combo
  - Esc :q!
    - Gets you out of the file without writing it.
- Where to go from here
  - *Learning the vi editor – Linda Lamb*
  - *vimtutor (under VIM)*
  - *http://www.jerrywang.net/vi/*

$ echo “Some Disk Commands”



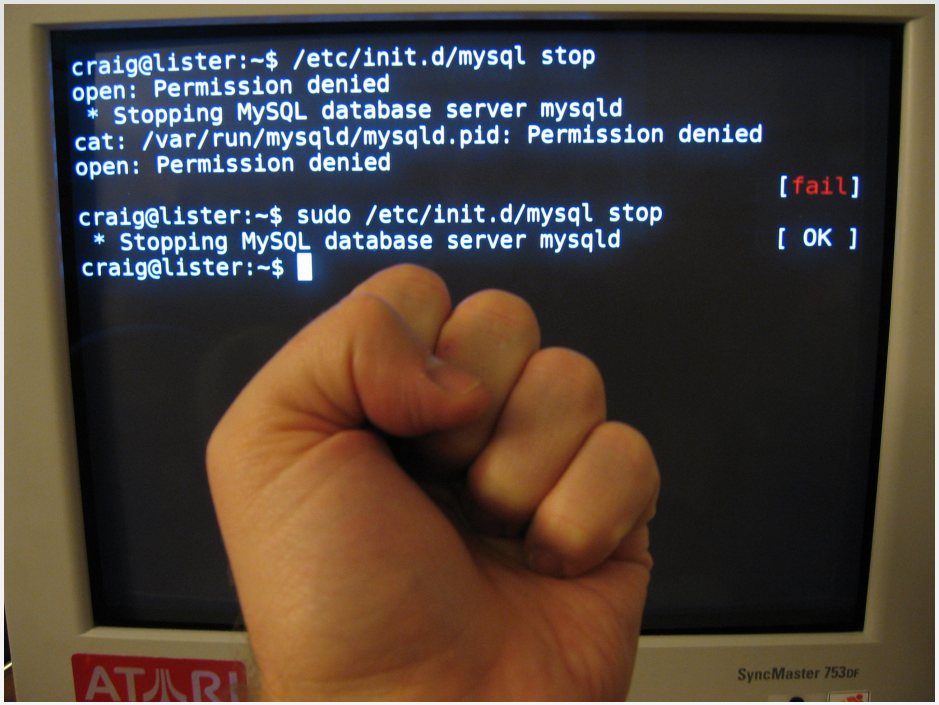
- Disk Space
  - `du` – Directory Usage
    - Traverses down a directory tree to show how much is being used
      - (Linux reports kilobytes, Solaris and others may report blocks. du -k reports kilobytes).
  - `df` – Disk Free
    - Shows mounted filesystems, and how much space is used.

$ echo “Who's online?”



- UNIX is multiuser, so how do we see who is online?
  - `w` – Nice formatted list
  - `who` – Functional
- Using `ps -ef` and `who`, you can tell what others are doing on the machine.
  - Useful, but be careful out there.

$ echo “root vs. the little people”



- UNIX has two classes of users, root and everybody else.
- Root has overriding permission to everything on the system – Be careful!
- sudo can be used to quickly get access and control access to privileged commands.

$ echo “Job Control”

- Job Control
  - Ctrl Z – pauses a job
  - `bg` – background
  - `fg` – bring a job to the foreground
  - `jobs` – list jobs currently running from the shell
  - Ctrl C – Cancels the running job
  - *`jobname &`* - same as Ctrl Z followed by 'bg'

$ echo “Permissions”

- Permissions
  - Three user types
    - User
    - Group
    - World
  - Three Permissions
    - Read
    - Write
    - Execute

$ echo “Sets of Permissions”

- Three roles for files / directories
  - User
    - Person who currently owns the file
  - Group
    - Set of users on the local system (workgroup)
  - World
    - Everyone else (not the user, and not in the group)

$ echo “Permissions (examples)”

- `rwxr-xr-x`
  - User has read / write / execute access (rwx)
  - Group has r and x
  - World has r and x
- `rwxr-x---`
  - User has r / w / x
  - Group has r and x
  - World has no access

$ echo “Pipes”

- Pipes
  - Understand these, and you'll understand UNIX.
  - Connect small programs into large operations.
- Example
  - `ps -ef | more`
    (paginate ps output)

$ echo “More about pipes”

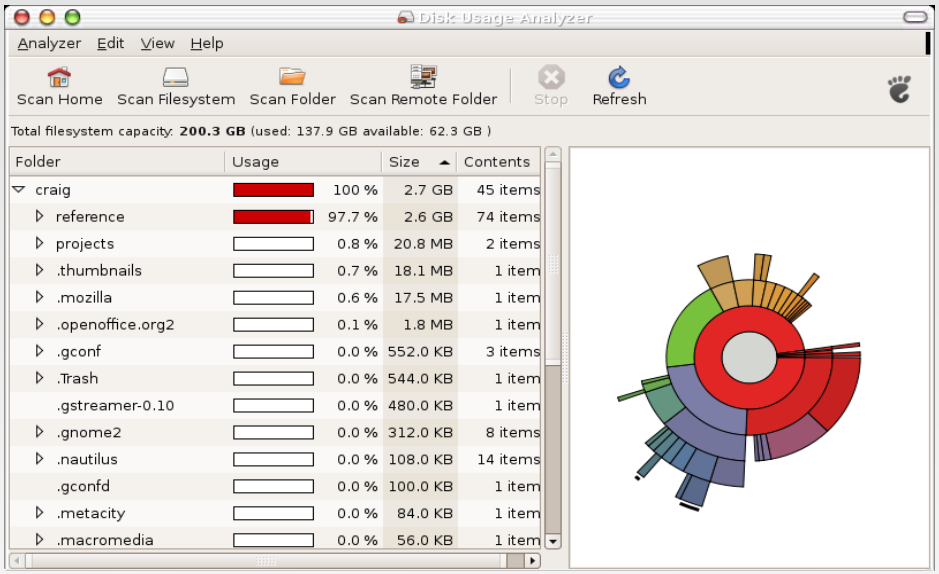
- STDIN / STDOUT
  - STDIN – The stuff you put in
  - STDOUT – The stuff you expect
  - STDERR – The stuff you don't expect
  - Most UNIX programs have support for STDIN/STDOUT

$ echo “Diagramming a pipe”

$ echo “More about pipes”

- More Examples
  - `ps -ef | grep craig`
    - finds all processes owned by me
  - `ps -ef | grep craig | grep -v ps | less`
    - finds all processes owned by me, omitting the ps command, and paginating the output
  - `cat file1 file2 > file3`
    - concatenate file1 and file 2 into file 3

$ echo “Productive pipes”

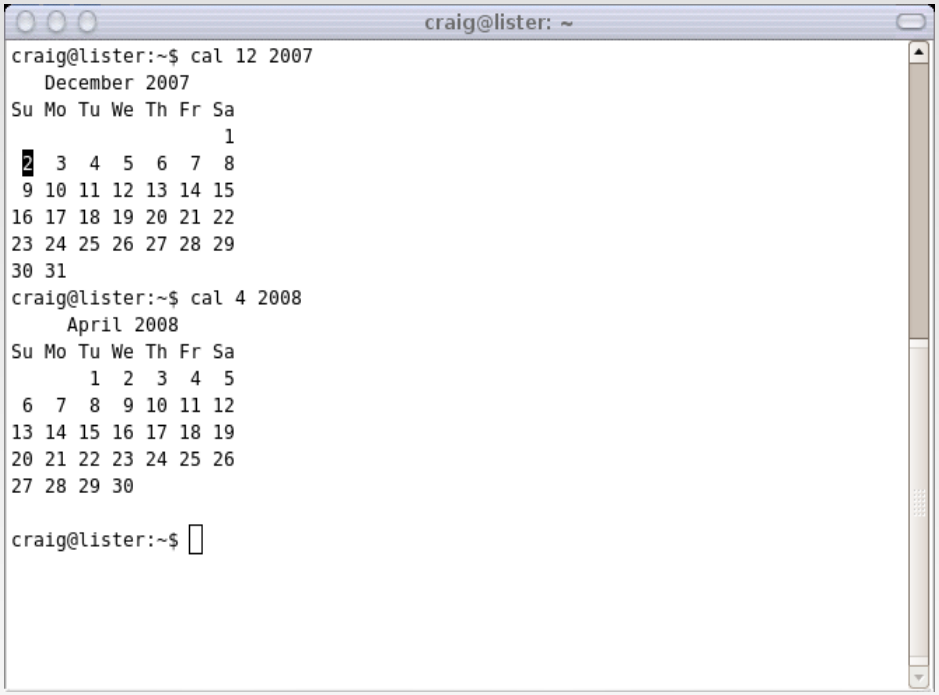


- Find out what is hogging up disk space:
  - `du -s * | sort -rn`
    - directory usage for all files in current directory only (-s *), pipe output to sort command, using reverse sort (-r), numeric rules (-n).
    - du | sort -rn will find directories taking up the most space

$echo “Productivity in UNIX”

- Most recent files in a directory:
  - `ls -ltr`
- Rename all files to .jpg using bash shell
  - `for $i in \`ls\`;do mv $i $i.jpg;done`
- Find all files with “craig” in them
  - `find . -exec grep \ -li craig {} \;`

$echo “Neat productivity tricks”

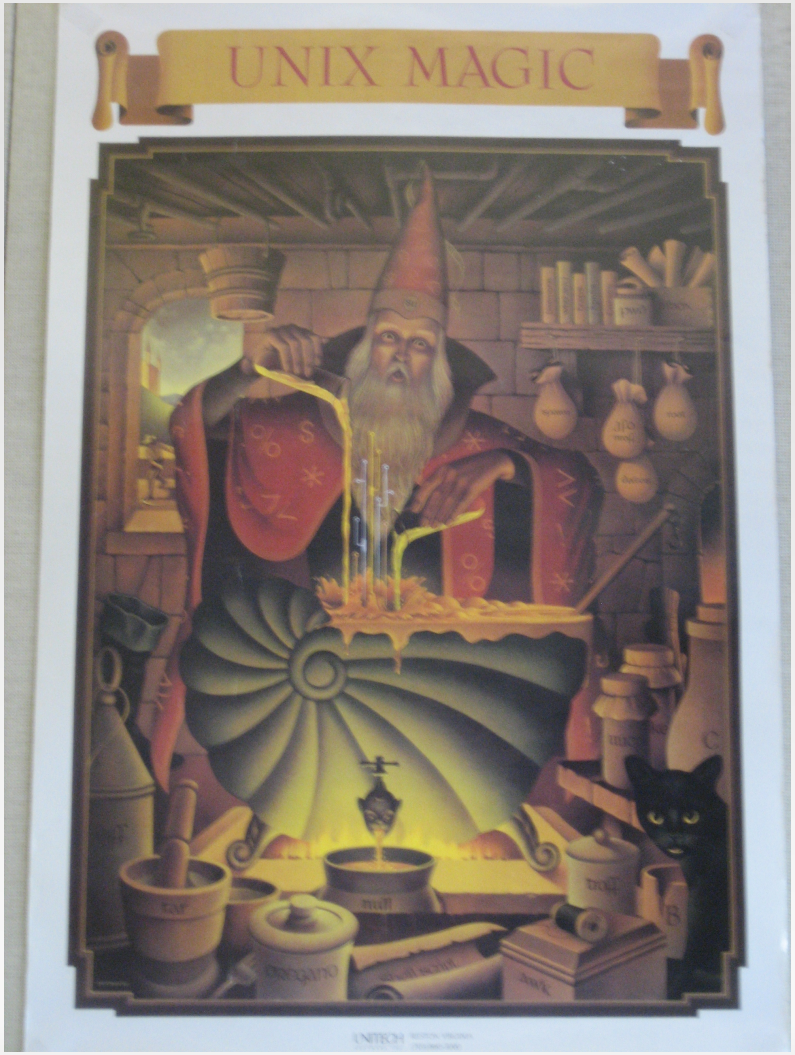


- Neat productivity tricks
  - `cal` – Shows calendar
  - `wc` – Shows word count (lines / words / bytes)
  - `diff a b` – Find differences between files named `a` and `b`
  - `bc` – interactive calculator

$echo “Yell at your admin if these aren't installed on your machine”

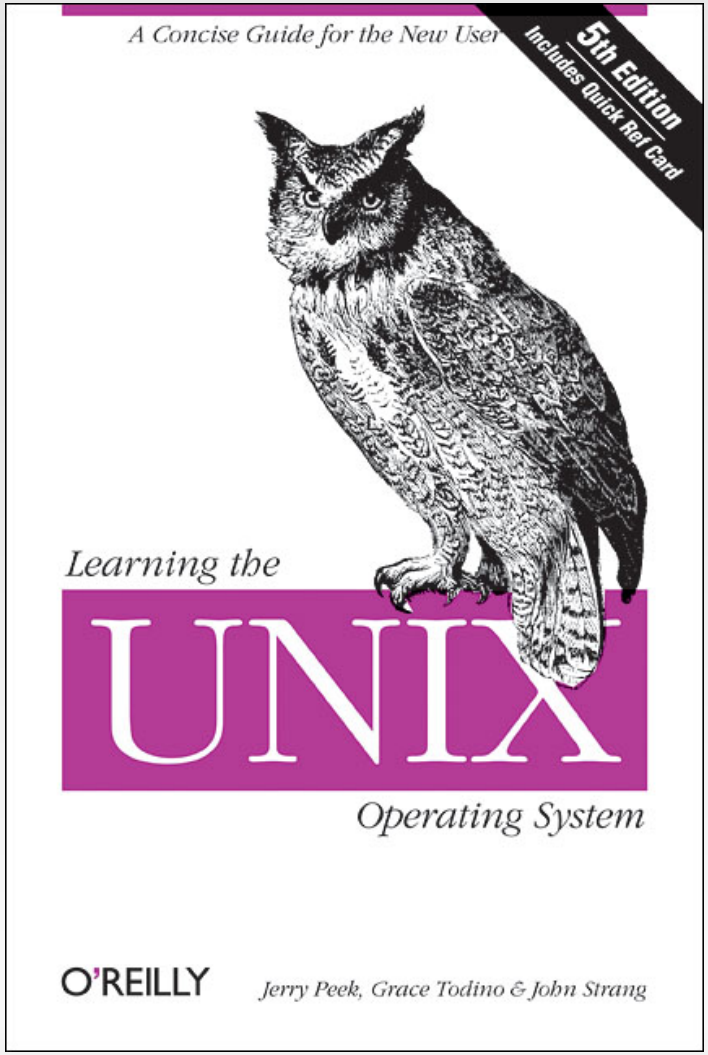
- `top` – REALLY useful process lister
- `locate` – Finds files by name in a hurry
- `whatis` – Shows a summary of what a command does
- `vim` – Vi IMproved – a MUCH better clone of `vi`.

$echo “Stuff we didn't cover”



- Stuff we didn't cover
  - `cut` – slices text files
  - `awk / sed (perl)`
  - Shell differences
  - `emacs` / other editors
  - Networking (`ssh / sftp / ftp / telnet`)
  - X protocol
  - `tar / cpio / dd`
  - A lot more...

$echo “Resources”



- Books
  - O'Reilly
    - Learning the UNIX O.S. 5th ed.
    - Running Linux 5th ed.
    - Linux in a Nutshell
- Websites
  - http://www.linuxcommand.org/
  - http://cb.vu/unixtoolbox.xhtml
  - http://www.pixelbeat.org/cmdline.html
- E-mail
  - craig@decafbad.net

$echo “Questions?”

$echo “Photo Credits”

http://www.flickr.com/photos/cobalt/1156232979/
http://www.flickr.com/photos/dannyman/9442491/
http://www.flickr.com/photos/laffy4k/43946201/
http://www.flickr.com/photos/l0b0/58141813/
http://www.flickr.com/photos/envios/93679057/
http://www.flickr.com/photos/formulaphoto/142471489/
http://www.flickr.com/photos/pitifulpussycat/145886769/
http://www.flickr.com/photos/samnewman/201940197/
http://www.flickr.com/photos/capecodcyclist/273440474/
http://www.flickr.com/photos/meeli/291326202/
http://www.flickr.com/photos/meeli/291326235/
http://www.flickr.com/photos/ninnghizidha/350870780/
http://www.flickr.com/photos/matsuyuki/369802424/
http://www.flickr.com/photos/telstar/370574085/
http://www.flickr.com/photos/auntiep/1939864/

$echo “More Photo Credits”

http://www.flickr.com/photos/extraketchup/408727666/
http://www.flickr.com/photos/twid/410697715/
http://www.flickr.com/photos/a440/443822837/
http://www.flickr.com/photos/cayusa/466865504/
http://www.flickr.com/photos/rcrowley/473535209/
http://www.flickr.com/photos/stevenerat/535050602/
http://www.flickr.com/photos/richardholden/568546230/
http://www.flickr.com/photos/mklingo/717372511/
http://www.flickr.com/photos/nfsw_photos/1331981828/
http://www.flickr.com/photos/jzawodn/227640085/
http://www.flickr.com/photos/harrywagner/6624437/
http://www.flickr.com/photos/coollibrarian/1087414098/
http://www.flickr.com/photos/the1pony/1431084418/
http://www.oreilly.com/catalog/lunix5/
http://decafbad.net/pictures/

$echo “Thank you!”

Thank you!

Contact Craig Maloney
e-mail: craig@decafbad.net
http://decafbad.net