UNIX Primer

Why care about unix

- availability: most industrial and academic environments use some version of unix
  - almost all web applications run on unix
  - mac osx is unix
- Flexibility: very customizable
  - easy to write and run code on it
  - Easy to network
- Price
  - many unix variants are open source (Linux)

Command Line

- main os interface for languages like dos
- Unix has GUIs (like mac and windows)
- More common to log in remotely to a command line
  - ssh, telnet
- on mac & ubuntu (unix variant): terminal
- on windows: cygwin emulates

File system

- unix uses a normal hierarchical file structure.
    - Directories - top directory is ‘/‘ rather than ‘C:\’
        - Major difference between unix and windows - forward vs. back slash
    - Files
- Extensions do not matter.
- Often used for convenience.
    - .py for python
    - .txt for text files
    - .csv for comma separated values (also a text file)

Relative vs. Absolute Path

- files can be referred to by their relative path
    - relative to the current directory
        - index.html
        - tmp/output.txt
- absolute to file system
    - /home/andrew/html/index.html

File System operations

- ls - list the contents of this directory
  - ls -l, ls -a
- mv - move a file to a directory, rename a file
- cp - copy a file to a directory, make a copy of a file
- cd - change directory (open a directory)
- mkdir - make a directory

Other commands

- rm - remove a file
- rmdir - remove a directory
- pwd - print working directory
- basename - remove leading directory from a file name
- Special characters
  - * - wildcard match everything. Similar to re.
  - .. - up one directory hierarchy
  - . - the current directory

in out err

- There are three main conduits for information to get in and out of programs.
    - Standard In “stdin”
        - any time you type information into a prompt, it goes via “standard in”
    - Standard Out “stdout”
        - print statements go to standard out
    - Standard Error “stderr”
        - error messages go to standard error

Reading files

- more, less - display a file one screen at a time
- cat - write the whole file to standard out
- echo <string> - write the string to standard out

Catching output

- python say_my_name.py
- python say_my_name.py > my_name.txt
    - Write standard output into a file
- python say_what.py 2> my_error.txt
    - Write standard error into a file
- python say_both.py &> my_out_and_error.txt
    - write both stderr and stdout to a file
- Appending — replace > with >>

Piping input

- “piping” Send the output of one program to the standard in to be read by another
    - The vertical bar | is referred to as a pipe

- cat my_name.txt | python say_hello.py > tmp.out
- echo “andrew” | python say_hello.py | python say_what.py 2> tmp.error
- also
    - python say_what.py < input_file > output_file

useful utilities

- These allow you to manipulate and query text files.
  - man
  - grep
  - wc
  - sed
  - cut
  - paste
  - sort
  - awk

man

- man <command> — manual entry for a command
- will tell you what “flags” you can send to a command and what the parameters are used for
  - Flags — e.g. “-l” “-v” are flags. these are special parameters that change how a command works or explain what to do with the following parameter

grep

- Run a regular expression against a file
  - one line at a time
- grep pattern file(s) — write lines that match the pattern to stdout
- grep -c pattern — count the number of lines that match
- grep -v pattern — write lines that don’t match
- grep -i pattern — case insensitive matching

wc du

- wc - word count
- Tabulates characters, lines and bytes in a file
  - -c -l -w, can specify only one of these to be written
- du -hs file - disk usage. How big is a file.

sed

- sed “stream editor” is a powerful tool that can do a lot with a single line.
- sed ’s/day/night/‘ - turns day (in stdin) to night (in stdout)
- can use any regular expression in this for replacement
- “sed one liners”
  - selectively print some lines
  - complicated string manipulation

cut

- cut -c 5-50 file - write the 5th through 50th character from each line

- cut -d “,” -f 5- file - use a delimiter “,” write the 5th through end of line fields.

- cut -d “:” -f 5,4,1 file - prints the 5th 4th and 1st fields

Appending: cat and paste

- Append files vertically
  - cat file1 file2 > output_file
- Append files horizontally
  - paste file1 file2 > output_file
  - paste -d ‘,’ file1 file2

sort uniq

- sort - sort in alphabetical order
- sort -n - Sort numeric order
- sort -r - Sort in reverse order

- uniq - write only unique lines (expects sorted input)
  - cat file | sort | uniq
  - -d - only duplicates
  - -u - only true uniques
  - -c - print the number of times seen

variables

- you can assign variables in the command line
- x=file.txt
- cat $x
- mv $x $x.data

variables

- remove an extension from a file
- x=file.txt
- basename $x .txt
- ${x%%.txt} — remove the .txt from the end
- ${x##prefix} — remove prefix from the front
- http://tldp.org/LDP/abs/html/refcards.html

for loop

- shell scripting - anything command you would type, you can put in a file and run as a “script”
    - this is like “programming unix”
    - simplest example

```
for x in *.txt; do
  python process_file.py $x > ${x%%txt}out.txt;
done
```