CCB 438/550, Spring 2022

Lecture 1: The high-performance computing environment

1 Course introduction

"lab course" lab is Amarel cluster
a) general intro to scientific computing
b) electronic structure c) MD simulation

2 A brief history of operating systems

a) Unix 70's Bell Labs → Linux
b) Microsoft MS-DOS → Windows 10, 11 → WSL
c) Apple Mac OSX derived from Unix, called Darwin
iOS app "ish"

3 The command line and file system

GUI vs CLI: terminal-based
jupyter-based
good for documenting, repeat
" remote access

$, >

folder aka directory
/home/<netid>/
/usr
/projects shared space on Amarel
/tmp

4 The "shell"

OS ↔ "shell" ↔ user interaction

/usr/bin/bash

① --help
-h
② man <command>

$ ls → list of files in the current folder
$ glb → shell look for a program called 'glb'

echo $PATH
/usr/bin:/usr/local/bin:/bin:
/home/dacase/bin

which glb

5 stdin/stdout/stderr and pipes

$ ls -l

default stdin is keyboard; stdout, stderr screen

$ ls > directory.txt create a new file with a list of filenames
< redirect stdin
2> redirect stderr

$ ls | wc ⇒ 6 17 256 on the screen
#lines #words #characters

→ $ ls | wc | awk '{print $1}' <ret>

FIFO

6 ~~Shell variables and programming~~

a) the various commands operate in parallel
wc command keeps reading its input until <EOF> is found

b) each program does one thing well
" " uses stdin, stdout, stderr
↳ combine individual programs with pipes

echo "ls | wc | awk {}" > count_files

$ bash count_files ⇒ $ countfiles

cf

CCB 438/550 Spring 2022

7 Introduction to AWK

→ awk '/pattern/ { action }' files to operate on

awk -f myprogam.awk

output to stdout

input from stdin

ls | wc | awk '{ print $1 }' 6 goes to stdout

↳ 6 17 250

$1 $2 $3

patterns: regular expression — syntax for finding patterns

→ a F 3 etc matches itself

. matches anything [except '\n']

^ $ " start/end of line

/^.a$/

[aei] matches a or e or i

[^aei] not (a or e or i)

/Case | Tom/ → special alteration = "or"

? zero or one of the preceding char

/aai?/ matches aai, aa

/abc?d/ " abcd or abd

* zero or any number of preceding char

/ */ any number of blanks

\+ 1 or more of preceding char

awk '/Etot/ { print $3 }' stdout just gets a list of total energies

actions: if, for(; ;) — init, ending condition, end-of-loop action

for (i=0 ; i<n ; i=i+1) { things to do }

BEGIN / END

...more AWK

BEGIN { statements }
/pattern/ { action }
END { final statements }

done on every line of input

- what is good/bad about AWK?
  - pros: simple to learn; great for "one-liners"; interpeted language
  - cons: slow for big things; cumbersome to write complex code
  - next step: python: pyscf; openMM; ML/AI torch
    compiled language: C/C++/Fortran
  - pro ~~neg~~ intuitive handling of strings
    look for two string functions
    - substr(string, start, length)
    - gsub(regex, substitute string, string)

Up next:

Databases DBMS

"personal database"

- only one person at a time
- modest amounts of data ~10 million
- modest complexity of the data