Introduction to Unix Computing in the Applied Mathematics Department

Chris Swierczewski

31 October 2013

Outline

1. ssh
2. Basic Unix Commands
   a. directory navigation
3. Intermediate Unix Commands
   a. utilities
   b. process management
   c. text editors
4. Advacned Unix Commands
   a. program redirection
   b. .bashrc
5. (Extras)

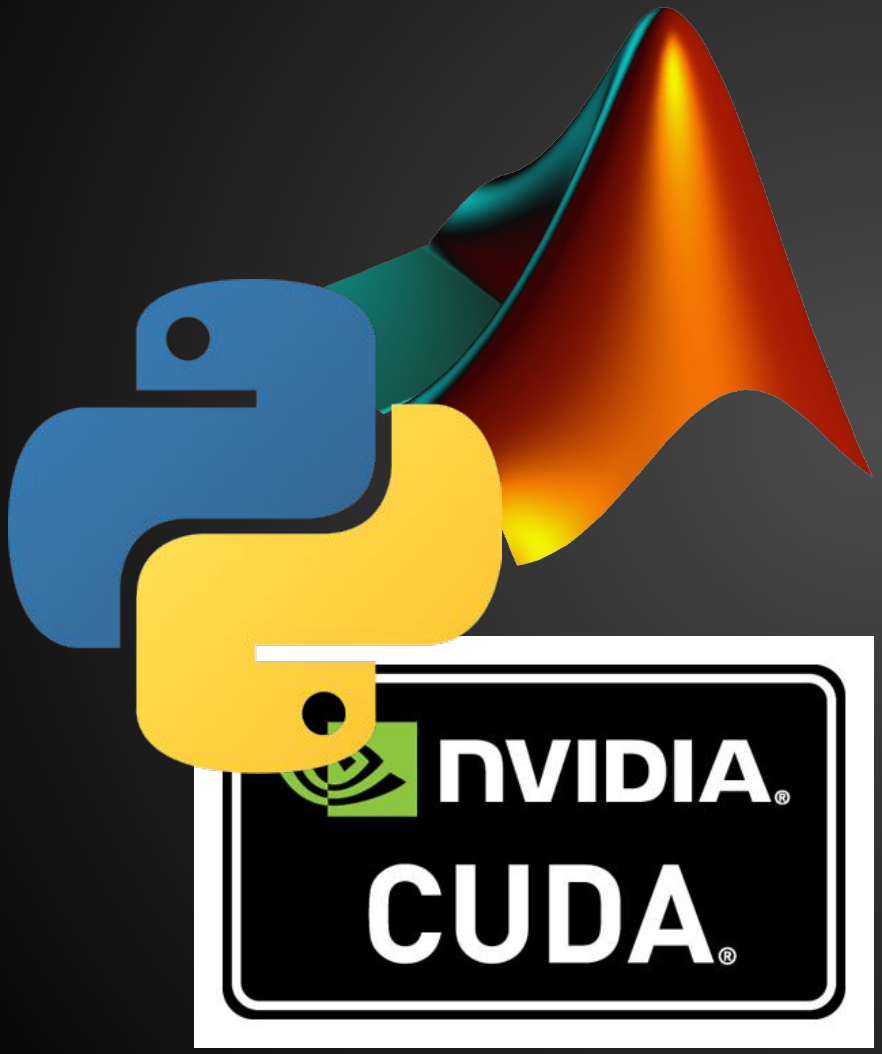
Our Equipment

- Linux Ubuntu 12.07 (non-milk coffees)
  - espresso.amath
    - x2 6-core Intel Xeon @ 3.07 GHz
    - L2 Cache (per core): 256 KB
    - L3 Cache (per proc): 12 MB
    - 24 GB RAM
    - NVIDIA Quatro 4000
      - 2 GB Device Global Memory
      - Double precision support
      - CUDA Compatible
  - americano.amath (coming soon!)

Our Equipment

- Apple OSX Lion (milk coffees)
  - latte.amath
  - mocha.amath
  - galao.amath
    - x2 6-core Intel Xeon @ 2.66 GHz
    - L2 Cache (per core): 256 KB
    - L3 Cache (per proc): 12 MB
    - 24 GB RAM
    - ATI Radeon HD 5870
      - 1GB Device Global Memory
      - Double precision support
      - Not CUDA Compatible
      - OpenCL Compatible

Installed Software



- **Mathematical**: Matlab, Maple, Python, Numpy, Scipy, matplotlib
- **Computational**: C/C++, FORTRAN, Python, CUDA (espresso only)
- **Logistic**: LaTeX, git, mercurial

Remotely Connecting

- via SSH (Secure Shell)
- OSX / Linux
  - Open Terminal
  - Type the following and hit 'Enter'

```
$ ssh myuwnetid@espresso.amath.washington.edu
```

  - Enter your UW NetID password and hit 'Enter'. (Note: password won't appear in terminal.)
- Windows
  - Download Google Chrome and Install “Secure Shell”

Demo...

(SSH'ing into espresso.amath)

A Quick Word About Dropbox

Use Dropbox to synchronize your files.

- OSX Instructions: (do the following once)
  - Physically log in to latte / mocha / galao
  - Setup using on-screen instructions
  - Lock screen (don't log out) by selecting the user list and scrolling down to "Lock Screen"
  - you can now ssh in and DB will run
- Linux Instructions
  - ssh into espresso / americano
  - run

```
$ dropbox start
```

(also: `$ dropbox autostart y`)

Basic Unix Commands

Unix-based computers are very common in math and science research. Get to know the Unix environment!

Resources:

- Using the Terminal
- Command Line
- Adv. Command Line

- `$ ls`
  - list directory contents
- `$ pwd`
  - print working/current directory
- `$ cd DIR`
  - change directory to DIR
  - "~" is home, "`..`" is parent
- `$ rm FILE`
  - remove / delete file. Cannot undo so be careful!
- `$ more FILE` or `$ less FILE`
  - view the contents of a file
- `$ top [-user username]`
  - view top CPU and memory using processes. Optionally, only those initiated / owned by `username`

Demo...

(basic terminal commands)

Intermediate Unix Commands

- `$ history`
  - shows past commands
- `$ man COMMAND`
  - manual page ("manpage") for COMMAND
  - use arrow keys or 'j' / 'k' to scroll
- `$ diff FILE1 FILE2`
  - compare two text files

File Management

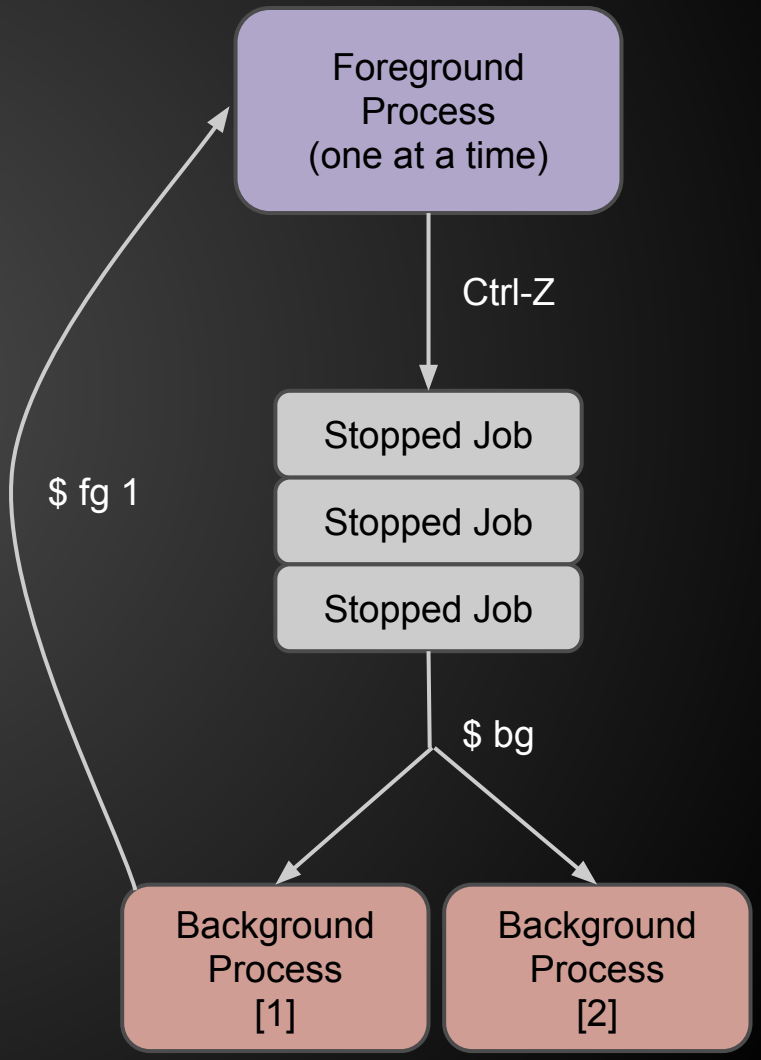
- `$ cp FILE DEST`
  - copy FILE to DEST
- `$ mv FILE DEST`
  - move FILE to DEST
- `$ du -sh DIR`
  - disk usage (hard drive space taken up) by DIR

Process Management

- `$ ps [-u USERNAME]`
  - list the processes started in this terminal session
  - (each process has a PID)
  - "`-u`" flag shows all processes owned by USERNAME
- `$ kill [-9] PID`
  - try to kill process PID
  - use the '`-9`' flag to force
- `$ nice -n7 COMMAND`
  - run COMMAND in a "nice" way. BE COURTEOUS TO YOUR FELLOW USERS!
- `$ renice +7 PID`
  - increase the "niceness" of process PID by 7 points

More Process Management

- Control-c
  - kill the currently running "foreground" process
- Control-z
  - stop (but don't kill) currently running foreground process
- `$ COMMAND &`
  - (include ampersand at end) runs COMMAND in the background allowing you to enter more commands
- `$ bg`
  - runs all stopped jobs in the background
- `$ jobs`
  - lists all background jobs
- `$ fg JOBNUMBER`
  - brings JOBNUMBER to the foreground



Demo...

(intermediate terminal commands)

Text Editors

Terminal-based text editors have a steep learning curve but greatly improve productivity.

- nano
  - easiest to use
  - tutorial: start nano and hit 'Control-g'
- vi / vim
  - tutorial: `$ vimtutor`
- emacs
  - tutorial: start emacs and hit 'Control-h' and then hit 't'.

Demo...

(opening text editor tutorials, editing files)

Intermediate Networking Commands

- Copy files from your computer to server

```
$ scp FILE myuwnetid@server:DIR/FILENAME
```

- `nohup`: execute COMMAND using `nohup` so it doesn't die when you disconnect. (output redirected to file)

```
$ nohup ping www.google.com &
```

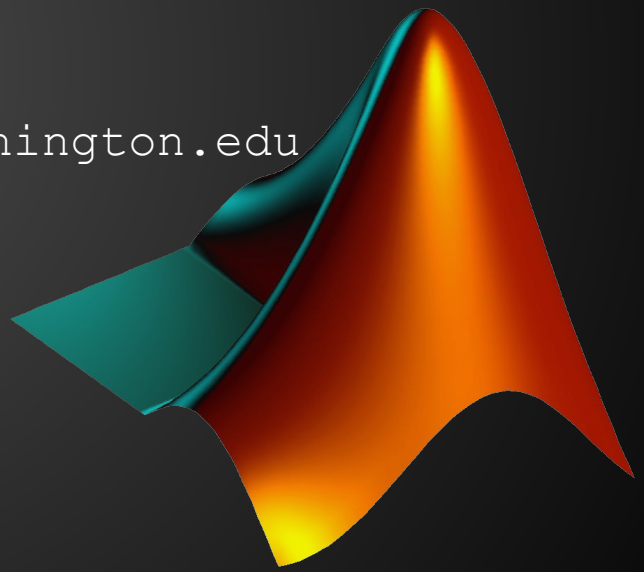
- `screen`: advanced tool for working with multiple terminals via ssh where, like in nohup, processes never die when you disconnect (they just fade away)

Matlab

- Cannot run Matlab interactively
  - requires X-windows forwarding. (High bandwidth.)

```
$ ssh -X user@server
```

```
$ ssh -X myuwnetid@latte.amath.washington.edu
$ matlab -nojvm
(Matlab startup)
>> plot([1,2,3],[-1,1,0],'g')
(plot is displayed)
>>
```



It's best to just run scripts that save plots to file and `scp` them back to your personal computer.

Demo...

(using Matlab via X-windows and `-nojvm`)

Advanced Unix Commands

Controlling Program Flow

- `$ COMMAND > FILE`
  - redirects output of COMMAND to FILE (overwrites if FILE exists)
- `$ COMMAND >> FILE`
  - appends output of COMMAND to end of FILE
- `$ COMMAND < FILE`
  - uses the contents of FILE as input to COMMAND
  - `$ sort < words.txt`
- `$ CMD1 | CMD2`
  - "pipes" the output of CMD1 to CMD2 as input
  - `$ ls | sort`
- You can string commands
  `$ ls | sort > sorted.txt`

Your .bashrc File

There is a special "hidden" file in your home directory called ".bashrc". Add terminal commands to the end and they will be executed every time you log in.

e.g. add

```
dropbox start
```

to start Dropbox upon logging in.

Other Tricks

```
alias ll='ls -l'
alias la='ls -a'

export PATH=~/bin:${PATH}
```

(add ~/bin to your "PATH" variable)

Demo...

(output redirection and a tour of .bashrc)

Questions?

Thank you!

Additional Tricks: Python and PDB

PDB (Python DeBugger) is a tool for stepping through your Python code line-by-line when debugging your code.

```
# script.py
import pdb
...
pdb.set_trace()
...
```

From command line:

```
$ python script.py
```

Navigation

`l`: (l)ook
`n`: (n)ext line
`s`: (s)tep into function
`c`: (c)ontinue to next breakpoint, error, or end of script

Additionally, you can execute Python commands from within the debugger. E.g.

```
(pdb) print x.norm()
```

Additional Tricks: screen

screen is a Unix tool for managing multiple "virtual terminal windows" within a single ssh session.

- no need to ssh twice so you can have two terminals open at once
- processes executed from screen will continue to run after you log off (as long as you "detach" screen instead of closing it)

Command List

- `$ screen`
  - start screen
- **Ctrl-a + d**
  - "detach" screen session (procs. will continue to run)
- `$ screen -r`
  - "resume" prev. detached screen session
- **Ctrl-a + c**
  - "create" a new screen window
- **Ctrl-a + "**
  - view a list of currently open screen windows
- **Ctrl-a + [0-9]**
  - switch to screen window number 0-9