# INTRO TO PROGRAMMING

# HELLO, CLASS

- I am: Ian McLaughlin ( im60@nyu.edu )
- Worked ~ 2.5 years at Next Jump
  - ecommerce
  - full stack
  - web dev
- Past ~ 1.5 years at Troveup
  - 3D printing
  - mostly frontend
  - graphics and modelling

# IMPORTANT RESOURCES

Course Device Minimum Specs:
8GB RAM and ~100GB free disk space

Course Public Github Repo:

https://github.com/pappasam/npd/tree/C1/c1_intro_programming

# VAGRANT

Vagrant is a tool for managing Virtual Machines (VMs)

- a VM is a simulation of computer hardware for you to install an Operating System  on plus whatever else
- The physical computer doing the simulating is called the *host*, while the VM is called the *guest*.
- Vagrant lets you pick from a public listing of pre-built OS images and define steps to install any needed dependencies with a surprisingly small amount of configuration

# VAGRANT: INSTALLATION

Download the installers and run them:

https://www.vagrantup.com/downloads.html

https://www.virtualbox.org/

# VAGRANT: WORKSPACE

Windows

(Run cmd.exe, optionally proceed from Desktop)

> cd Desktop

Apple

(Run Terminal, should already be in home directory)

(for either OS)

> mkdir vagrant-workspace

> cd vagrant-workspace

> vagrant init hashicorp/precise64

> vagrant up

# VAGRANT: CONNECT

You will need to have an ssh accessible from your command line to let Vagrant handle connecting:

- Windows: Install ssh through cygwin
- Apple: use ssh to connect
- Linux: if not ssh not already available, install package ssh-client

```
# the following starts an ssh client directly in the terminal
$ vagrant ssh
```

# VAGRANT: ALT-CONNECT

You can use any ssh client on your host machine to connect to the VM, which by default has the following credentials:

username: vagrant

password: vagrant

Or the windows client Putty can also be configured to use the auto generated ssh key:

https://github.com/Varying-Vagrant-Vagrants/VVV/wiki/Connect-to-Your-Vagrant-Virtual-Machine-with-PuTTY)

# WELCOME TO YOUR VM

```
Using username "vagrant".
vagrant@127.0.0.1's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_6

 * Documentation:  https://help.ubuntu.com/
New release '14.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Fri Sep 14 06:23:18 2012 from 10.0.2.2
vagrant@precise64:~$
```

That last line is the *command prompt*, which tells you the *shell* is ready for you to type new commands.

Complete setup instructions from file `a1_complete_vimtutor.md` if you get here early.

# VIM

- One of the most common tasks of a programmer is manipulating text
- your primary text editor is a deeply personal choice, but for the purposes of this course we'll be using Vim
- commands needed to install vim within virtualbox
  - sudo apt-get update --fix-missing
  - sudo apt-get install vim
- has builtin tutorial, after installing run command 'vimtutor'

# VIM CONFIG

```
$ cd ~
$ wget https://raw.githubusercontent.com/boombador/dotfiles/master/vi
$ mv vimrc .vimrc
```

# TERMINAL

- a program that provides a 'shell', which is an interface based on text commands to interact with operating system functionality and resources like the file system
- Windows has *cmd*, mac has *Terminal*, and linux usually has *xterm* if you're in a graphical environment

## First Commands

cd - change directory

ls - list directory contents (`dir` in windows)

mkdir - create a new directory

# COMMAND SHORTCUTS

- Certain symbols are shorthand for several
  - `~` - (tilde symbol) home directory
  - `.` - (dot/period) current working directory
  - `..` - (double dot) parent directory
- Examples
  - cat ~/.bashrc
  - ls ..
  - cd ~/../..

# HOME SWEET HOME

- OS generally creates a 'home' directory containing a subdirectory for each user on the system which by default only they and admin users can access
  - user im60 would have home dir at `/home/im60`
  - this is like a Windows user's desktop, though linux graphical environments may also create a `/home/im60/Desktop` directory
- your initial working directory when logging into a system will be the home directory
- you can store configuration, programs, and files here

# FILE SYSTEMS

- an organized method for representing and operating on data on a storage device
  - persistent storage
  - volatile memory
- can allocate portions of disk into *partitions* which can use different file systems
- Many types and versions
  - FAT/FAT32, NTFS
  - ext2/ext3/ext4
  - zfs
  - initramfs

# FS ORGANIZATION

- `/` - the *root* directory that contains everything else
- `/bin` - essential command binaries
- `/boot` - static files used to OS
- `/dev` - device files representing hardware status or IO interfaces
- `/mnt` - Mount point for mounting a filesystem temporarily
- `/usr` - Secondary hierarchy
- `/var` - Variable data

FHS: https://d37djvu3ytnwxt.cloudfront.net/asset-v1:LinuxFoundationX+LFS101x+1T2016+type@asset+block/LFS101_Ch3_Sec1_FSH.pdf

# FS ADDRESSING

- File location is represented by a string identifier known as a *path*
- Possible to dynamically generate lists of paths for more expressive commands
- two ways to interpret paths:
  - **absolute -** the unambiguous location of the file with respect to the file system root
    `cat /boot/grub/grub.cfg`
  - **relative -** takes into account current location context to omit parts of the absolute path
    `cd /boot/grub ; cat grub.cfg`

# FILE SYSTEMS: RELATED READING

- https://help.ubuntu.com/community/LinuxFilesystemsExplained
- http://www.cyberciti.biz/tips/understanding-unixlinux-file-system-part-i.html
- http://www.tldp.org/LDP/intro-linux/html/sect_03_01.html

# SOME BASIC COMMANDS

- cat - write the contents of files specified via argument to STDOUT
- echo - write arguments to STDOUT
- which - print path of executable that would be executed by argument

# IO REDIRECTION

- `< filename.txt` - (less than character) redirect contents of `filename.txt` to STDIN of previous command
- `>` - (greater than character) redirect STDOUT of previous command to a file
- `|` - (pipe character) redirect STDOUT of previous command to STDIN of next

```
grep oldVariable < file_source.txt
cat file1.txt > copy1.txt
cat shell_script.sh | bash
```

# USING GIT

- Git is one of the most popular *Version Control Systems* (VCS)
- Common to want to 'go back in time' to see how a file used to look or track changes, with VCS instead of saving multiple files for each version the system stores a record of changes
- Collection of this project historical data is referred to as the *repository*

# SVN HISTORY

## OPTIONAL

# BEFORE GIT: SVN

- Subversion is another popular VCS that came before git, considered less sophisticated but easier to understand
- Centralized, meaning all contributors communicate with a single authoritative SVN server storing the whole repo
- The project files on developer's machine (or VM) are called the *working copy*, which are downloaded from the central server

# SVN CONTINUED

- If you use a text editor to modify a file, the svn tool indicates your version has *local changes* that haven't been stored on the central server
- You can submit these changes to the central server by *committing* them, which submits the file changes to the central server
- This set of changes is referred to as a commit, and also includes a user written summary of the changes plus a randomly generated unique identifier to reference the commit later

# GIT: START

- Use the `git init` two part command to create an empty repository
- repository is just the `.git` directory, files from project are reconstructed by git program
- Nothing is in the repository after you create it

```
$ git init
Initialized empty Git repository in /home/im60/example/.gi
$
```

# GIT: STATUS

- `git status` command displays a helpful summary of files in the working directory
- also suggests commands, depending on context, which we'll go over presently

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   instructions.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committe
  (use "git checkout -- <file>..." to discard changes in w

        modified:   changelog.txt
$
```

# GIT: STAGING

- When you make changes to a file in a git-controlled directory, the repo hasn't changed
- You can *stage* files with the `git add` command, causing git to simulate adding the changes to the persistent history log

```
$ echo "add a line to modify" >> file.txt
$ git add file.txt
```

# GIT: COMMIT

- Commit takes all staged changes and makes an entry in the history that can be referenced by the automatically generated unique commit ID

```
$ git commit -m "add a bugfix line in setup"
[C1 9175f99] add a bugfix line in setup
 1 file changed, 3 insertions(+), 2 deletions(-)
$
```

# GIT: BRANCH

- A branch is a reference to a commit
- Usually kept updated to point to latest in line of commits, most recent point of development

```
$ git branch
  bugfix-02
  release-qa
* master

$ git branch new-feature
$ git checkout new-feature

$ git branch
  bugfix-02
  release-qa
  master
* new-feature

$
```

# GIT: MERGE

- Merging a branch into another adds all the commits from the first into the second
- Either by updating branch references, or creating a new commit that incorporates changes from both branches

```
$ git branch
  bugfix-02
  release-qa
  master
* new-feature

# add changes from new-feature branch to master branch
$ git checkout master
$ git merge new-feature
```

# GIT: CLONE

- Copies a repository, can be used to download from another server or copy a repo on disk
- Cloning adds default *remote* named 'origin'

```
 $ git clone https://github.com/pappasam/npd.git
Cloning into 'npd'...
remote: Counting objects: 273, done.
remote: Compressing objects: 100% (204/204), done.
remote: Total 273 (delta 91), reused 0 (delta 0), pack-reused 6
Receiving objects: 100% (273/273), 887.41 KiB | 702.00 KiB/s, d
Resolving deltas: 100% (120/120), done.
Checking connectivity... done.
 $
```

# FOR NEXT CLASS

## Assignment 1:

https://github.com/pappasam/npd/blob/C1/c1_intro_programming/assignments/a1_complete_vimtutor.md

## Class 2 Handout:

https://github.com/pappasam/npd/blob/C1/c1_intro_programming/handouts/lesson_plan_students_c01_s02.pdf

## Useful Linux Overview (focus on chapters 3, 6, 7, 8):

https://www.edx.org/course/introduction-linux-linuxfoundationx-lfs101x-0