Git/Unix Lab

March 2015

Version Control

- Keep track of changes to a project
- Serves as a backup
- Revert to previous version
- Work on the same files concurrently
- ***Must-know in the tech industry***

Origins of VC

- Concurrent Versions System (CVS)
  - Released in 1986
  - Predecessor of Subversion
- Darcs Advanced Revision Control System
  - Released in 2003
  - Predecessor of Git

Version Control

- 3 widely used tools:
  - Git (highly recommended)
  - Subversion (meh)
  - Mercurial (barely used)
- 2 main hosting websites (for git):
  - GitHub (5 private repos upon request)
  - Bitbucket (unlimited private repos)

Note: Cornell requires you to use **private** repos for homeworks & projects.

Bitbucket - Follow Along

To do right now:

- Create or log into your Bitbucket account
- Create a repository named CS3410-Unix

Git - Basics

- `git add <filename>`
  - Tells git to record the changes you made to the file
    - creation / deletion / modification
  - git *stages* these changes
  - `git add README.md #Add changes in README`
- `git commit`
  - Tells git to bundle the changes together into a *commit*
  - Add a commit message with `-m`
  - `git commit -m “Changed README.md”`

Git - Follow Along

- Open your terminal
- `$> git clone <bitbucket path>`
- `$> cd CS3410-Unix`
- Create netid.txt
- Write your netid in netid.txt

Git - Follow Along

- `$> git add netid.txt`
- `$> git commit -m “Added netid.txt”`

Git - Basics

- `git push <remote> <branch>`
  - Pushes your local commits to your online repo
  - <remote> is the url of your online repository
    - `origin` variable holds that url
  - <branch> will be `master` for now
  - `git push origin master`
- `git log`
  - Summary of all commits and their messages

Git - Follow Along

- `$> git push origin master`

Git - Basics

- `git pull <remote> <branch>`
  - <remote> usually `origin`
  - <branch> is `master` for now
  - `git pull origin master # fetch changes made`
    `# by someone else`

Git - Useful Tip

- `git stash`
  - Temporarily store your local changes
  - Makes sure you do not have conflicts when pulling
  - `git stash pop #Reapply stashed changes`

Git - Basics

A typical sequence of instructions you will use:

```
$> git stash #Store local changes
$> git pull origin master #Fetch changes from repo
$> git stash pop #Reapply local changes
$> git add -A . #-A also adds deletions
$> git commit -m “Changed file x.txt” #Commit stage
$> git push origin master #Push changes to repo
```

Shell Scripting

Shell Scripting

- As you have probably noticed, the CS3410 course VM has **no GUI.**
- We interact with a ***command line interface*** in order to tell the computer what we want it to do.

Shell Scripting

- Through the shell you can:

- Run programs.
- Interact with the file system.
- Change settings.
- Send/receive e-mails.

Bash

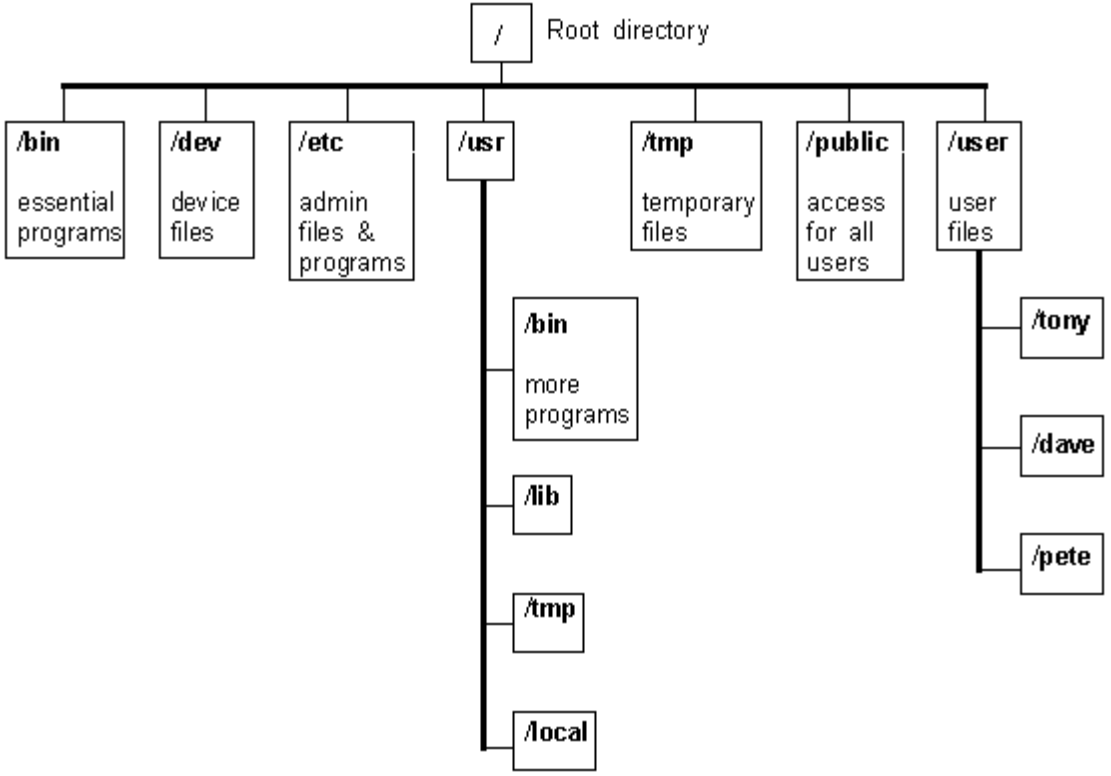
- **Bash** Unix shell is the default shell on Linux and Mac OS X.
- There are many (many, many) commands. Here we will present the most common/useful.

The File System

- Every file and directory has a path!
- Path: Where the file or directory is located in the file system.
- Unix paths are delimited by forward slashes “ / ”
  e.g. /home/username/cs3410/pa1

Special Directories

- **Root Directory**
- The Top-Most dir!
- “ / ”
- **Home Directory**
- Current User’s dir!
- “ ~ ”
- **Current Directory**
- The dir you’re in!
- “ . ”
- **Parent Directory**
- The dir above!
- “ .. ”



Special Directories

- **Absolute paths**
- start with “ / ”, i.e the root directory

- **Relative paths**
- start from your current directory

File System Commands

- `cd <dirpath>` : Change Directory
- `pwd` : Print Working Directory
- `ls` : List Directory contents
- `mkdir <dirname>` : Make Directory

- When you log in to VM you are in your Home Directory!
- You can always access it using “ ~ ”

File System Commands

```
$ pwd
/home/username
$ ls
dir1  file1
dir2  file2
$ mkdir mydir
$ ls
dir1  mydir file1
dir2  file1
```

```
$ cd ..
$ pwd
/home
$ cd ~/mydir
$ pwd
/home/username/mydir
$ cd /home
$ pwd
/home
```

Create & Delete Commands

- `mkdir <dirname>`: Make Directory
- `touch <filename>`: Create an empty file
- `rm <filename>`: Remove the file
- `rm -r <dirname>`: Remove dir and files in it recursively

- WARNING: There is no Trash in the Unix File System. If you delete a file or directory, it is gone forever!

Create & Delete Commands

```
$ pwd
/home/username
$ ls
dir1  mydir file2
dir2  file1
$ touch myfile
$ ls
dir1  mydir file2
dir2  file1 myfile
```

```
$ rm myfile
$ ls
dir1  mydir file2
dir2  file1
$ rm -r mydir
$ ls
dir1  file1
dir2  file2
```

Command Options

- `rm -r <dirname>`

  -r makes the rm command run recursively!

  -r is an option that changes the command!

- **How to know the command options and how to use them?**

The most helpful command

- `man <cmd>`
  - Opens the **man**ual page for the command *cmd*
  - Man page includes: Usage, Description, Explanation

- If the man page is confusing, you can always try googling the problem!

Command History

- All commands you run are saved!

  ```
  $ history
  ```

- Cycle through previous commands with the arrow keys
- Very helpful when executing a small set of commands frequently (e.g. “nano arraylist.c” “gcc arraylist.c”)

Tab completion

- Pressing the tab key will automatically complete whatever you are typing
- If there is more than one thing you could be typing (so tab completion will not work), press tab twice to see the list of possibilities

Redirection

```
$ command > file
```

- Send the output of the command to that file.
  - Creates the file if it does not exist.

```
$ ls ~ > homefiles.txt
```

- Writes list of files under Home Directory to homefiles.txt

Redirection

`$ command > file`

- Will overwrite the contents of file

`$ command >> file`

- Will append the output of command to *file*

Shell scripting

- You can write programs to do all the things you want in the Unix shell!

- A Shell script is a bunch of commands saved in one executable file.
- Uses extension .sh

Count files in Directory

```
#!/bin/bash

COUNTER=0

for i in $( ls ); do
     COUNTER=$[$COUNTER+1];
done

echo $COUNTER
```

- Use bash interpreter
- Initialize Counter
- Count
- Print

Shell scripting

- The Shell file has to be executable!
- Write script, name filename.sh
- Make it executable and execute!

- `$ chmod +x filename.sh`
  - will make file executable
- `$ ./filename.sh`
  - execute that file
  - requires “.” to be used explicitly! (You should google why!)