Intro to Linux & Command Line

Based on slides from CSE 391

Edited by Andrew Hu

slides created by Marty Stepp, modified by Jessica Miller & Ruth Anderson

http://www.cs.washington.edu/391/

Lecture summary

- Unix and Linux operating system
  - You will not be tested
  - Just to help you understand what the environment is

- Introduction to Bash shell (command line)

Operating systems

- What is an OS? Why have one?
- Ever heard any of these words? Linux, Unix, GNU

Linux

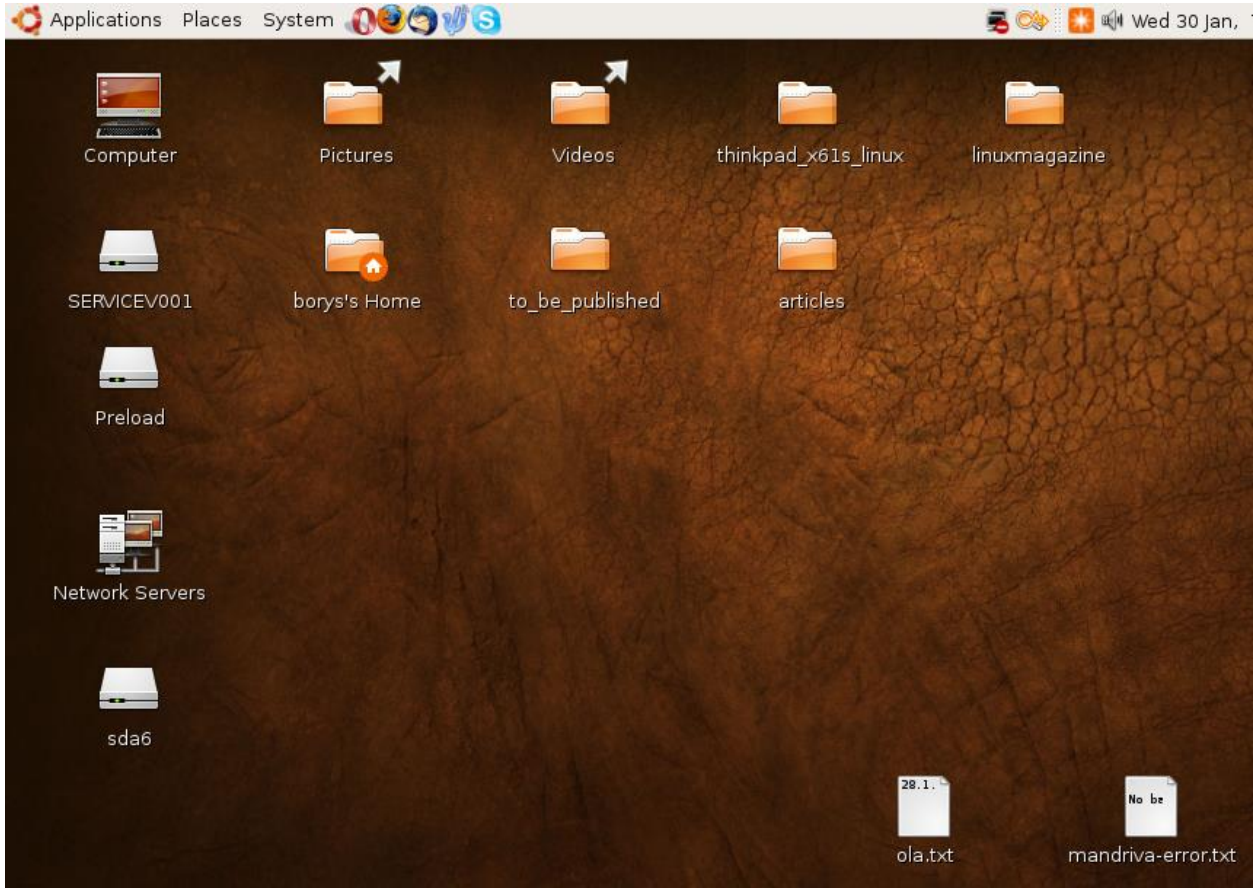
- **Linux**: An OS framework
  - Almost every server on the internet
  - Every Android phone
- **GNU**: A bunch of small tools
- **Distribution (distro)**: Different Linux types
  - e.g. Ubuntu, Debian, **Raspbian**
- Key features of Linux:
  - **open source software**: source can be downloaded
  - free to use
  - constantly being improved/updated by the community

Linux vs Unix

- What the heck is a Unix?
  - Early OS, the progenitor of Linux
  - Nobody really uses Unix anymore
- Many people use them interchangeably
  - http://unix.stackexchange.com
- Don’t worry about it

Linux Desktop

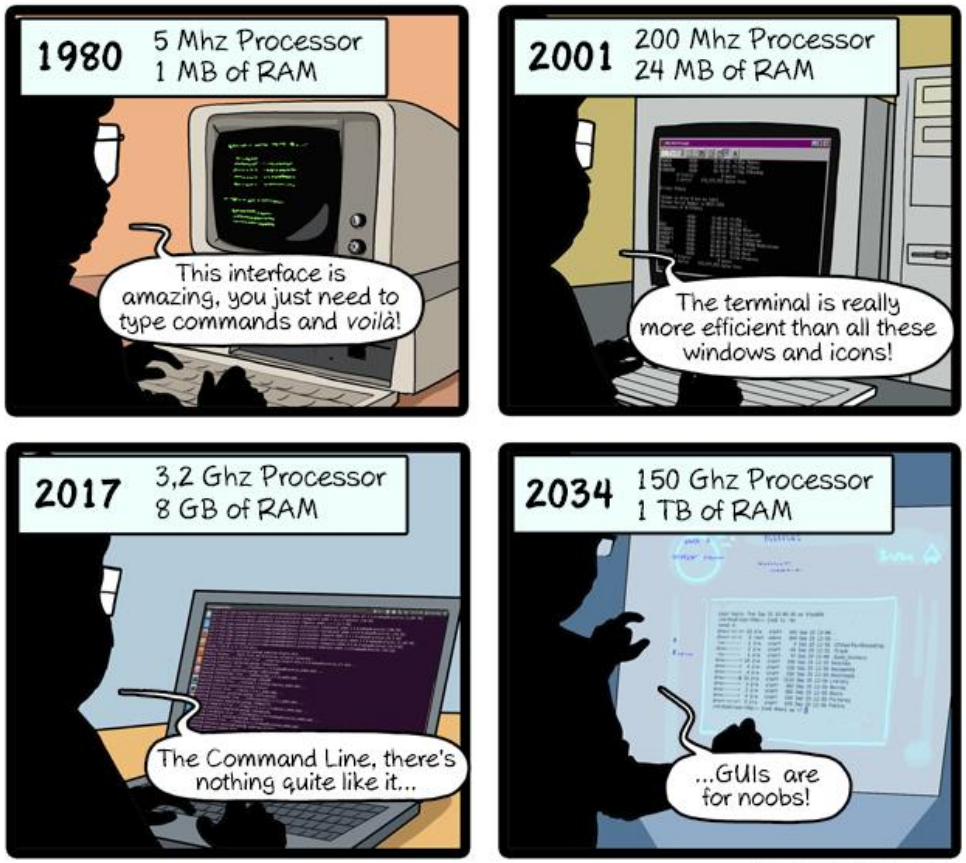
- Very similar to navigating Windows or OS X



Things you can do in Linux

- Browse the internet
- Install and play games
- Play Music and Videos
- IM, Skype
- Basically: Everything

Command Line



CommitStrip.com

Shell

- **shell**: uses user input to manage the execution of other programs.
  - Runs in a text window
  - User types commands, the shell runs the commands

- We will use Bash
  - Most commonly used shell on Linux

- Why should I learn to use a shell when GUIs exist?

Git Bash: /d/Projects/avladov/idea-vue-templates

```
Angel@ANGEL-W ~
$ cd /d/Projects/

Angel@ANGEL-W /d/Projects
$ gulp watch
[00:34:23] Local gulp not found in D:\Projects
[00:34:23] Try running: npm install gulp

Angel@ANGEL-W /d/Projects
$ cd avladov

Angel@ANGEL-W /d/Projects/avladov
$ cd idea-vue-templates/

Angel@ANGEL-W /d/Projects/avladov/idea-vue-templates (master)
$ ls
README.md  build.bat  extract.bat  settings/  settings.jar  src/

Angel@ANGEL-W /d/Projects/avladov/idea-vue-templates (master)
$ build
```

Why use a shell?

- Why should I learn to use a shell when GUIs exist?
  - faster
  - **work remotely**
  - programmable
  - **can handle repeated commands**

Navigating the file system

`ls`	lists files in a directory
`ls -l`	lists all files in directory with details
`cd [dir]`	**c**hanges the working **d**irectory

GUI (you don't see this)

Me

CSE

Docs

Notes

Shell (what you see)

```
$ ls
```

Navigating the file system

`ls`	lists files in a directory
`ls -l`	lists all files in directory with details
`cd [dir]`	**c**hanges the working **d**irectory

GUI (you don't see this)

Me

CSE

Docs

Notes

Shell (what you see)

```
$ ls
Me CSE Docs Notes
$
```

Navigating the file system

`ls`	lists files in a directory
`ls -l`	lists all files in directory with details
`cd [dir]`	**c**hanges the working **d**irectory

GUI (you don't see this)

Me

CSE

Docs

Notes

Shell (what you see)

```
$ ls
Me CSE Docs Notes
$ cd CSE
$
```

Navigating the file system

`ls`	lists files in a directory
`ls -l`	lists all files in directory with details
`cd [dir]`	**c**hanges the working **d**irectory

GUI (you don't see this)

142

143

391

Shell (what you see)

```
$  ls

Me CSE Docs Notes

$  cd CSE

$
```

Directory commands

command	description
`ls`	list files in this directory
`ls -all`	list all files in this directory *with details*
`ls [dir]`	list files in a given directory
`pwd`	**p**rint the current **w**orking **d**irectory
`cd [dir]`	**c**hanges the working **d**irectory
`mkdir [dir]`	create a new directory
`rmdir [dir]`	delete a directory (must be empty)

- some commands (`cd`, `exit`) are part of the shell ("builtins")
- others (`ls`, `mkdir`) are separate programs the shell runs

Relative directories

directory	description
`.`	the directory you are in ("working directory")
`..`	the parent of the working directory (`../..` is grandparent, etc.)
`*`	everything in a directory
`[string1]*[string2]`	everything in a directory that starts and ends with the specified strings
`*.java`	everything ending with the `.java` suffix
`~`	your **home** directory (on many systems, this is `/home/`***`username`***)

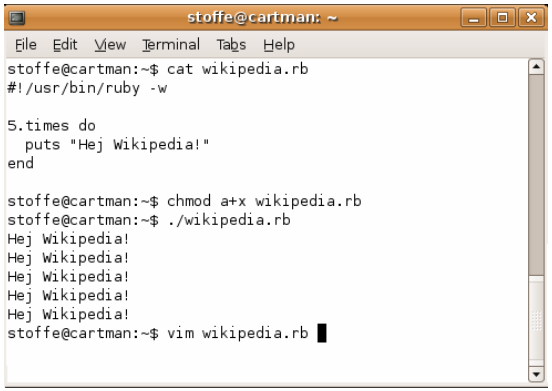
File commands

command	description
`cat [file]`	print out the contents of file (named for con**cat**enate)
`pico [file]`	edit a file in the working directory
(while running) CTRL+C	stops the currently running program
`wget [url]`	download a file from the internet
`alias short='long'`	create a short version of a long command

- CTRL is often written as ^
  - CTRL+C would be ^C

Shell commands

- many accept **arguments** or **parameters**
  - example: cp (copy) accepts a source and destination file path



- a program uses 3 streams of information
  - **input**: comes from user's keyboard
  - **output**: goes to console
  - **errors** can also be printed (by default, sent to console like output)

- parameters vs. input
  - *parameters*: before Enter is pressed; sent in by shell
  - *input*: after Enter is pressed; sent in by user

Command-line arguments

- Options: hyphen '-' followed by a letter
  - `gcc program.c –o program.o`
    - `-o = "output"`
  - some are longer words preceded by two `-` signs, such as `--count`
- many programs accept a `--help` or `-help` option to give more information about that command (in addition to `man` pages)
  - or if you run the program with no arguments, it may print help info

File commands

command	description
`cp`	copy a file
`mv`	move or rename a file
`rm`	delete a file
`rm –r [DIR]`	delete a directory and all of its contents
`touch`	create a new empty file, or update its last-modified time stamp

- Use "-i" (interactive) to force confirmation
  - `$ rm -r –i *`

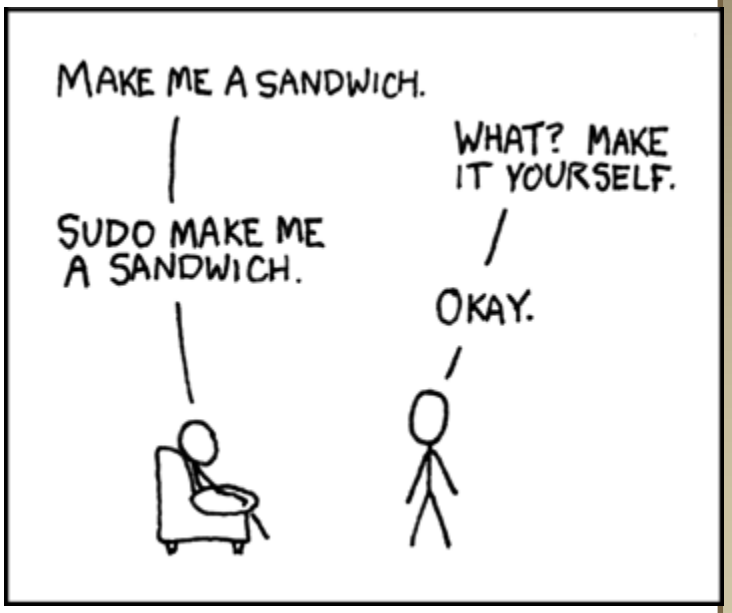
    `remove file 'bitcoinwallet.key'? no`
- `touch prog.java –t 149204010000`
  - `-t [[CC]YY]MMDDhhmm`

SUDO

- **s**uper **u**ser **do**
  - Pronounced like “sue-dough” or “pseudo”

- similar to “execute as Admin” in Windows

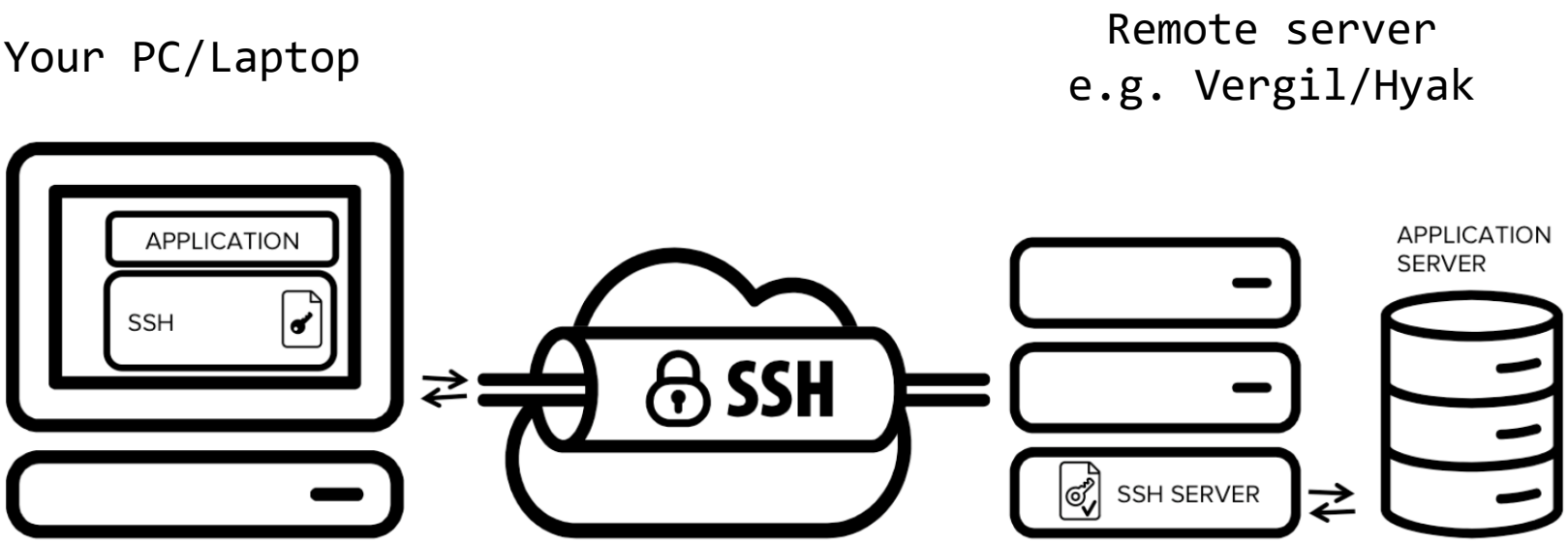
- `sudo apt-get install [package]`
- `sudo apt-get update`
- `sudo rm [someone else’s file]`



SSH

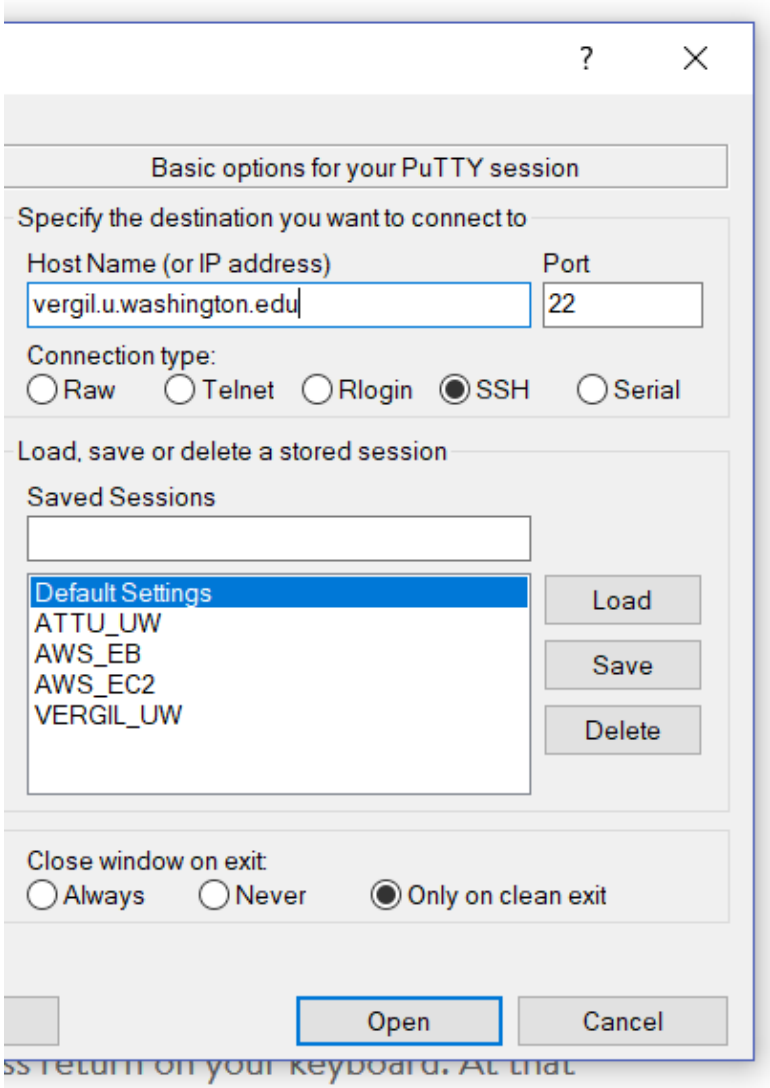
- **S**ecure **Sh**ell

- Control a computer remotely, through a shell
- Do work from the comfort of your personal computer



SSH Clients

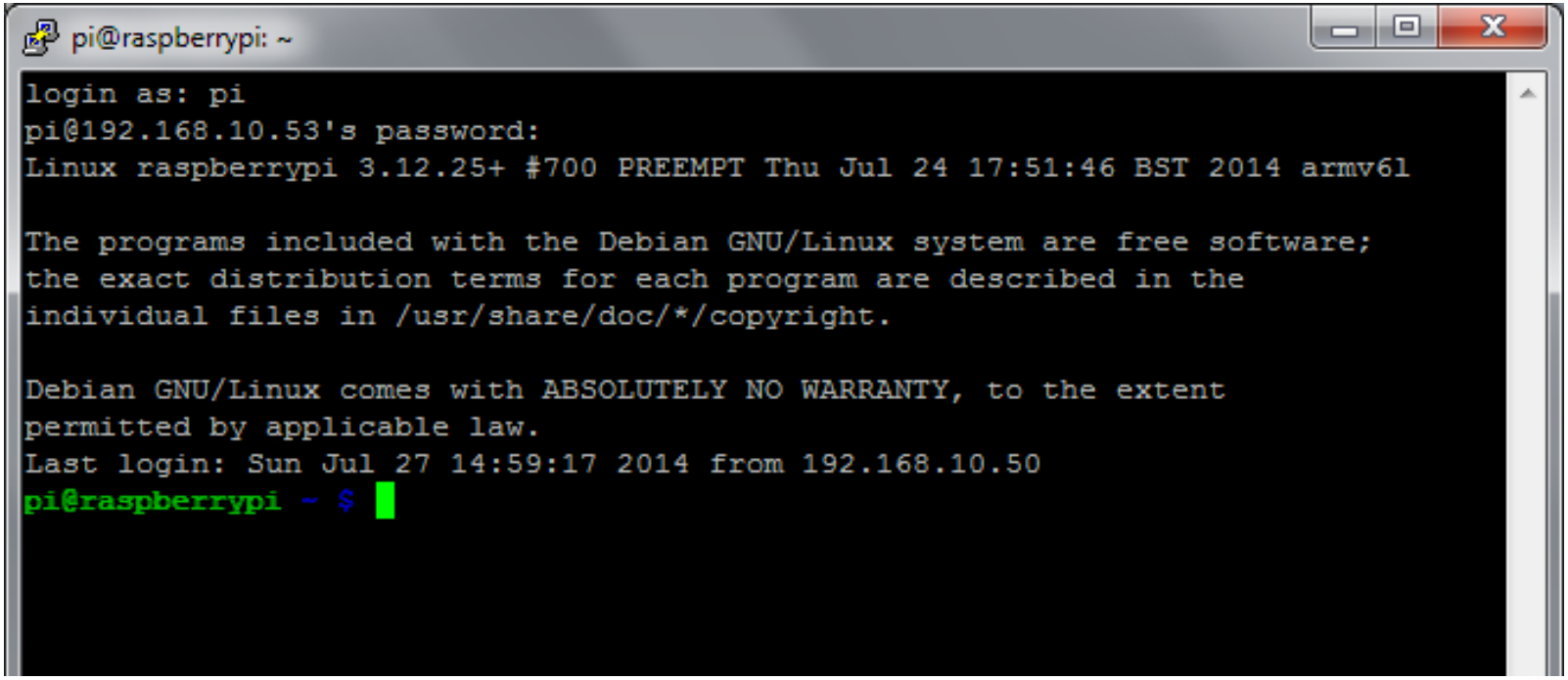
- Windows: PuTTY
  - type in server URL
- Mac: Built in shell client
  - just type `ssh user@example.com`
- Port should be 22



Using SSH

- Shell interface

- This is why it's important to know how to use command line!



How to collaborate on code?

What have you tried in the past?

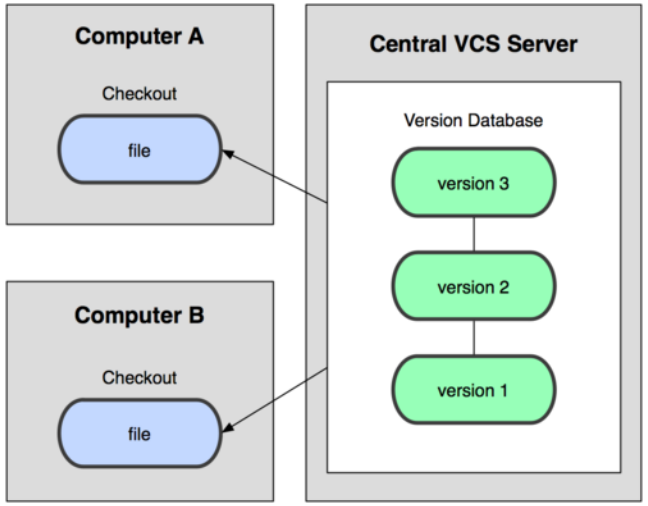
Version Control

- When you work on a group project, how do you share files?
  - Google Docs, OneDrive, Dropbox
  - These are examples of real-time centralized version control
- What happens when two people try to edit the same line?
- What happens when somebody deletes everything?
- How do you know who made what changes, and when?

Real-Time Centralized

- You're probably very familiar with using a real-time centralized Version Control System
  - E.g. Google Docs, OneDrive, Dropbox

- Pros
  - Easy to set up, use, and share
  - Instant feedback from other editors
- Cons
  - Can't really resolve conflicts
  - Other editors "stepping on your toes"
  - You don't know who, when, or what each change was



What is a conflict?

```
print("The meaning of life is...");
```

=

```
print("The meaning of life is...");
print("The meaning of life is 42");
```

```
print("The meaning of life is...");
print("The meaning of life is that it ends");
```

What is a conflict?

```
print("The meaning of life is 42");
```

≠ CONFLICT

```
print("The meaning of life is...");
print("The meaning of life is that it ends");
```

What is a conflict?

```
print("The meaning of life is 42");
```

OR

```
print("The meaning of life is that it ends");
```

What is a conflict?

```
print(“The meaning of life is 42”);
```

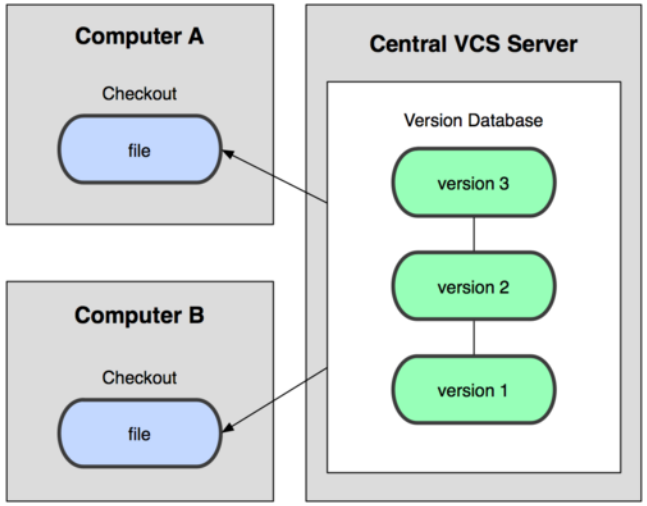
```
print(“The meaning of life is that it ends”);
```

What is a conflict?

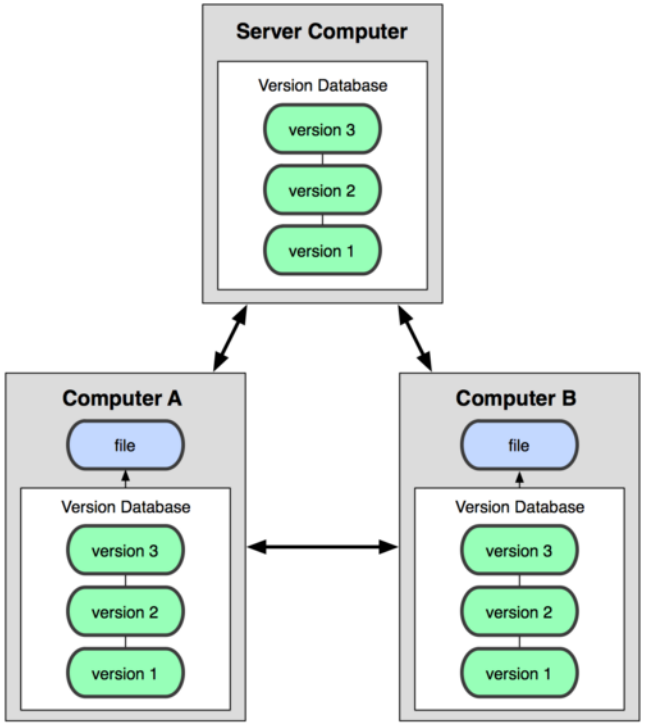
```
print(“The meaning of life is that it ends”);
```

Distributed VCS

Centralized Model



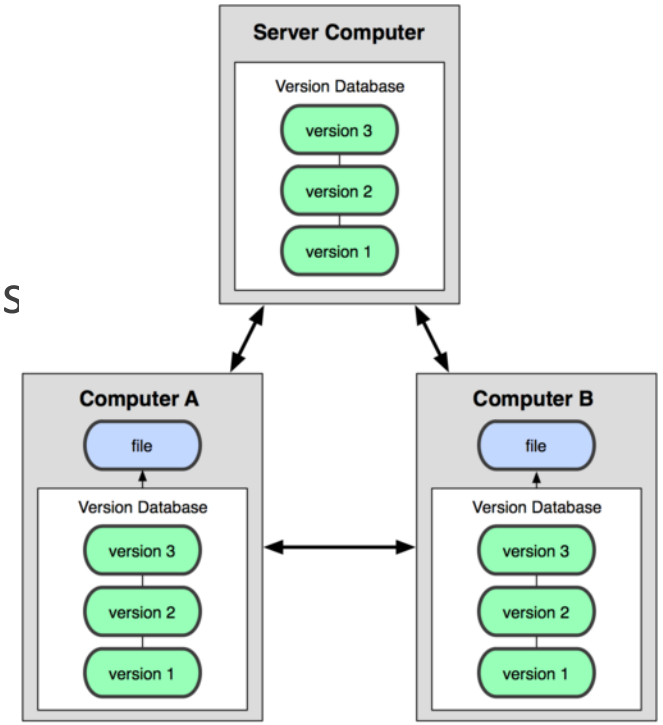
Distributed Model



Distributed VCS

- A distributed VCS lets each user have their own copy of the database (called a repository or “repo”)
- Users can send their versions to each other
  - If there is a conflict, the sender decides which version to use

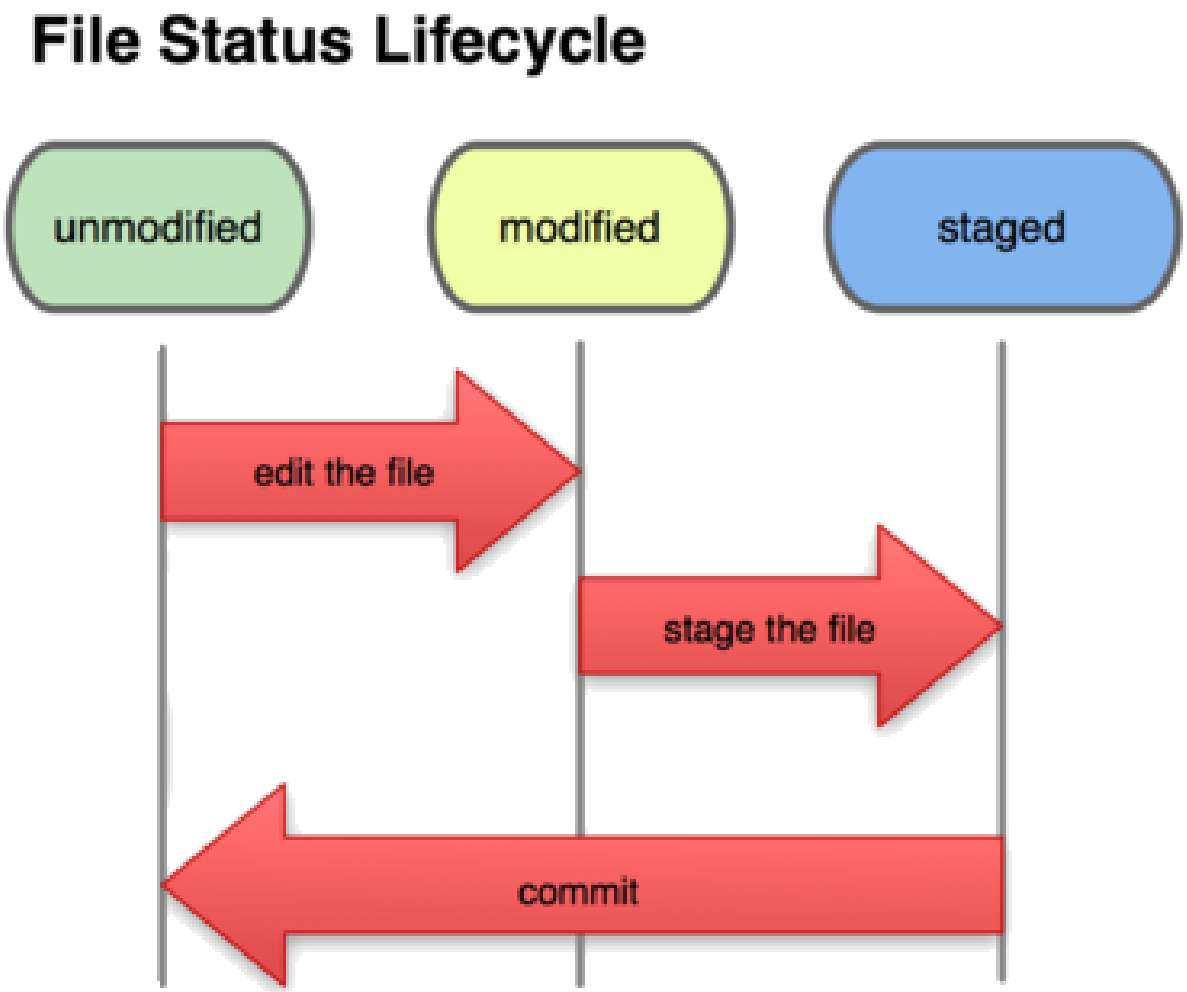
- Pros
  - Each user’s repo is equally valid
  - Users can choose how to resolve conflicts
  - Detailed log of every change ever
- Cons
  - More difficult to set up and use



Git

- Git is a distributed VCS
- Most commonly used VCS
- Created by Linus Torvalds to support the development of Linux

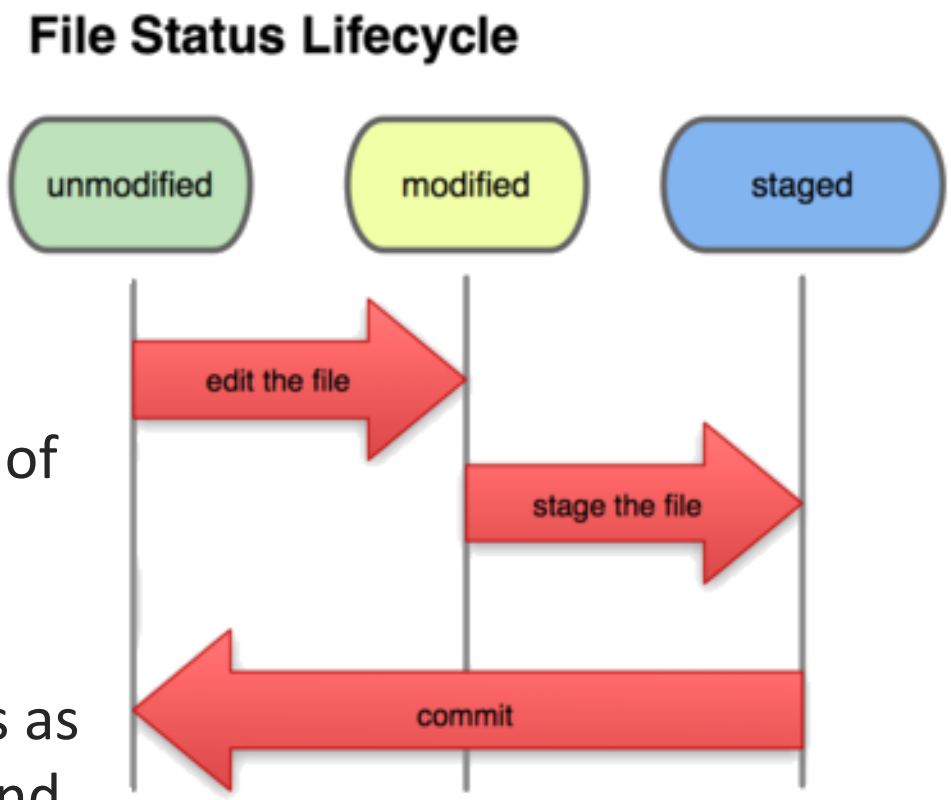
Git on your local machine



Basic Workflow

Basic Git workflow:

1. **Edit** files in your working directory
2. **Stage** files, adding snapshots of them to your staging area
3. **Commit**, which takes the files as they are in the staging area and stores that snapshot permanently



Use Good Commit Messages

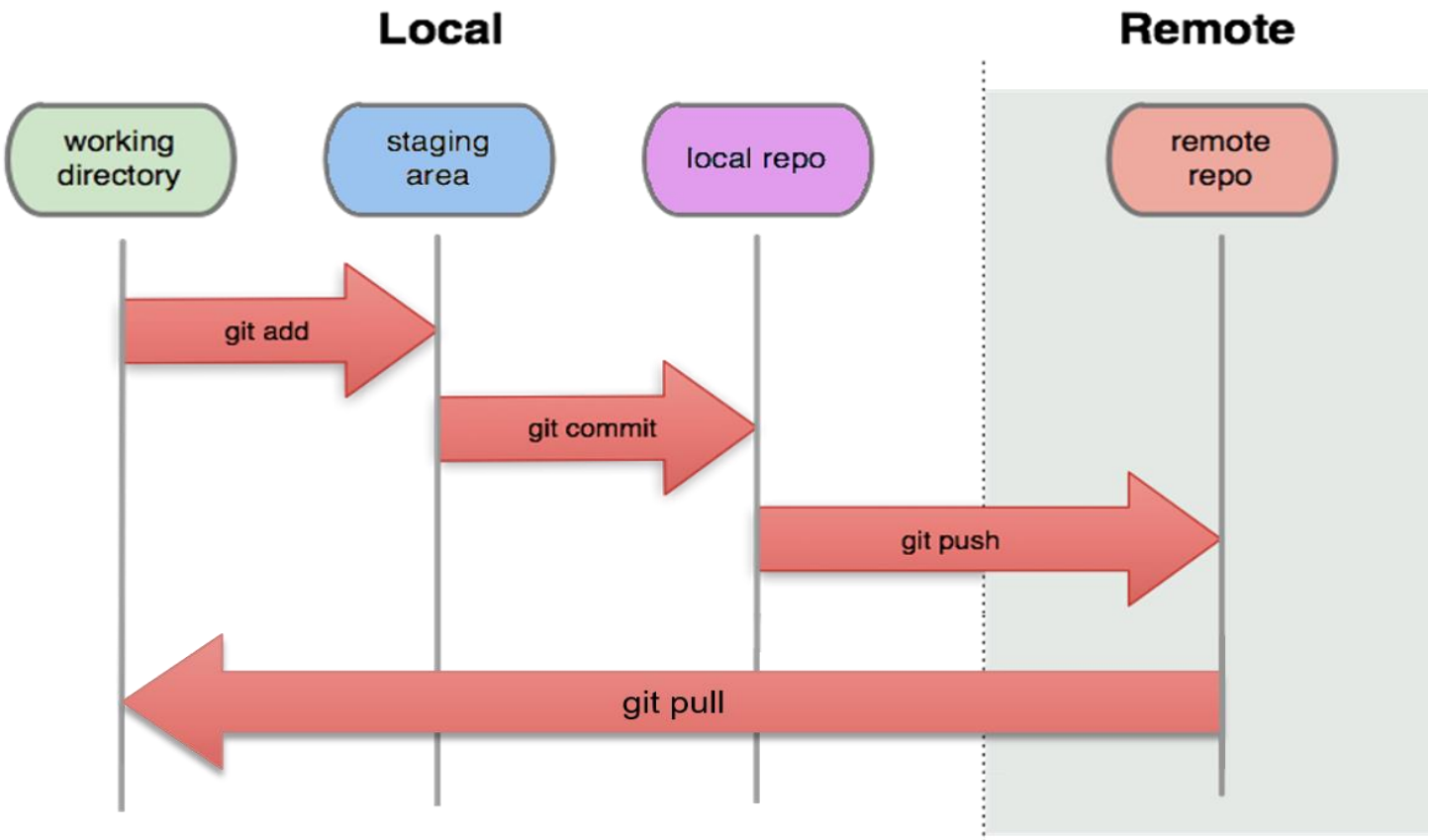
HTTP://XKCD.COM/1296/

Commit message style

- Start your commit message with either +,-,=
  - + when functionality was added
  - - when functionality was taken away
  - = for small edits that don't change the functionality
- Less than one sentence
  - You can usually omit language like "added" or "removed" since it's implied by the + or -
- Use multiple commits as needed
  - If you added two separate features, make two separate commits!

When in doubt,
Commit more, rather than less

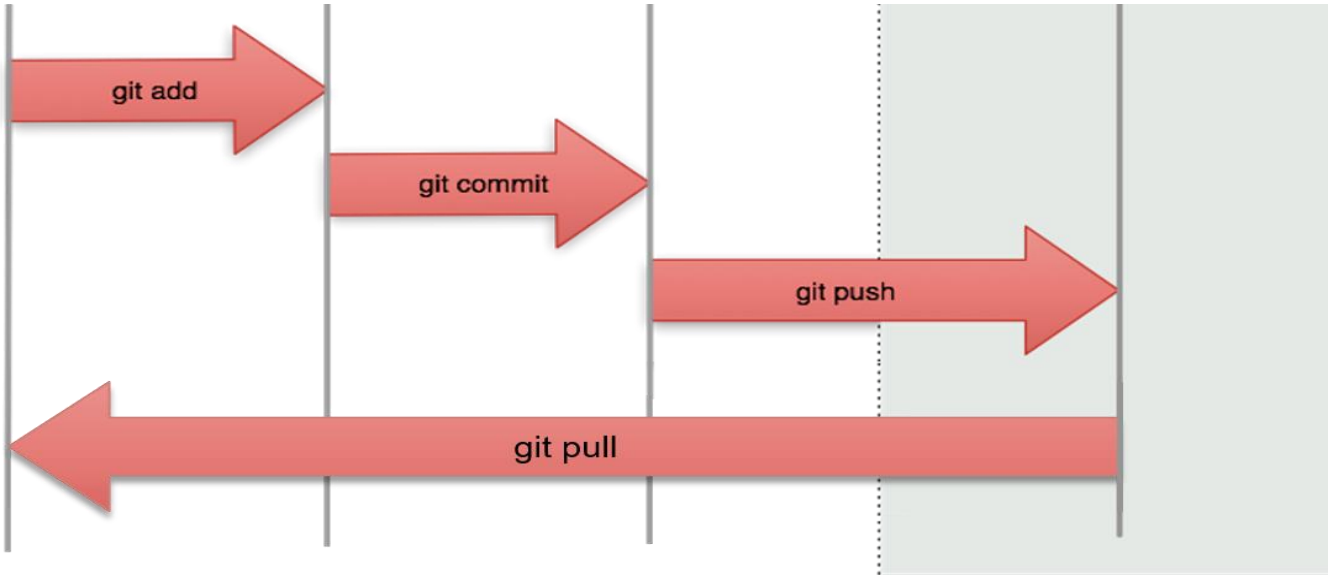
Git with a remote server



- **Push** commits to the server to publish your commits
- **Pull** from the server to get other users' commits

Git shell commands

command	description
`git status`	See which files are/need to be staged
`git add [-A] [file]`	Add `[file]` to the staging area
`git commit –m [mssg]`	Commit these changes and label `[mssg]`
`git push [origin master]`	Send your commits to the server
`git pull`	Download updates from the server



Vi / Vim

- Powerful command line text editor
- Notoriously infuriating
- [“how to exit vim” is one of the most commonly asked questions on Stack Overflow]

Basic Vim Commands

command	description
`:w`	Write changes to the file
`:x`	Write and quit
`:q!`	Quit without saving
`i or a`	Enter Insert mode (AKA normal typing mode)
`v`	Enter Visual mode (AKA select mode)
ESC	Exit Insert or Visual mode

https://wiki.gentoo.org/wiki/Vim/Guide and http://tnerual.eriogerg.free.fr/vimqrc.pdf