CS 197U: Introduction to Unix

Lecture 5: Compression, Mounting, and Package Managers

Instructor: Arun Dunna

Lectures: Monday/Wednesday, 4pm - 5:15pm, LGRC A301

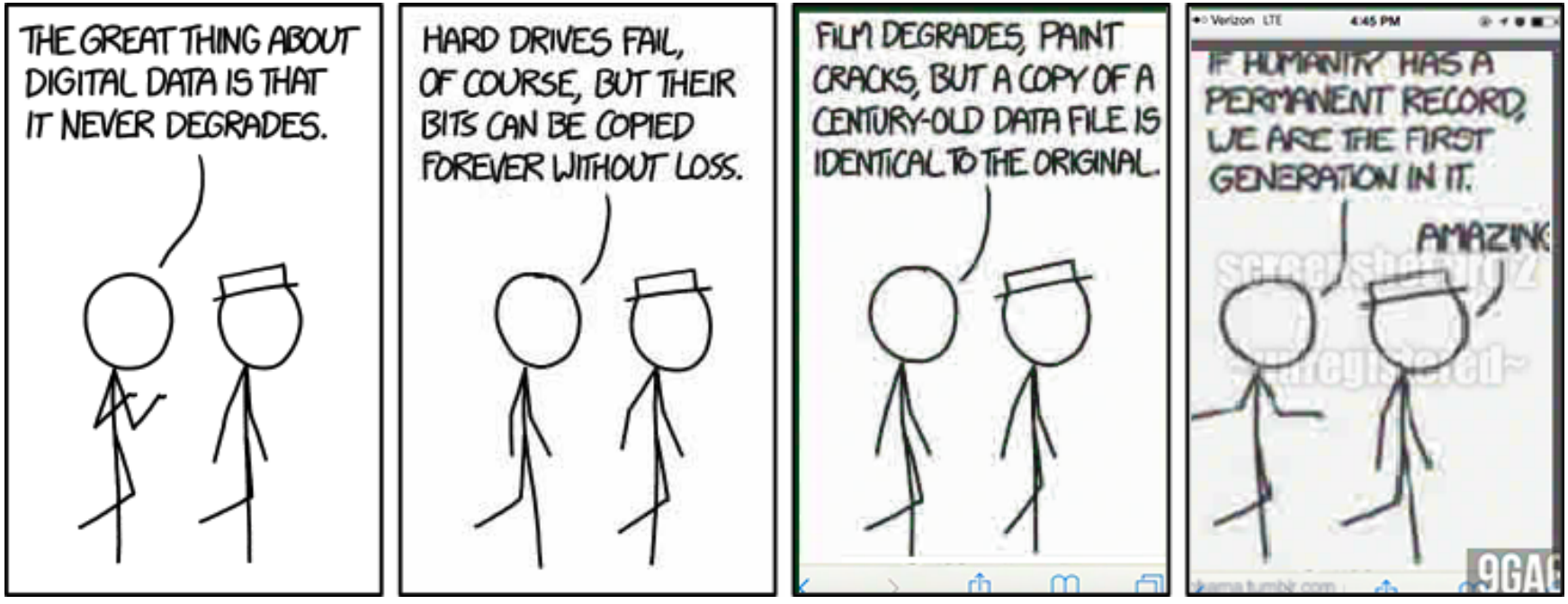
Lab 3 / Quiz 3

Due Sunday, 2/17, at 11:59pm

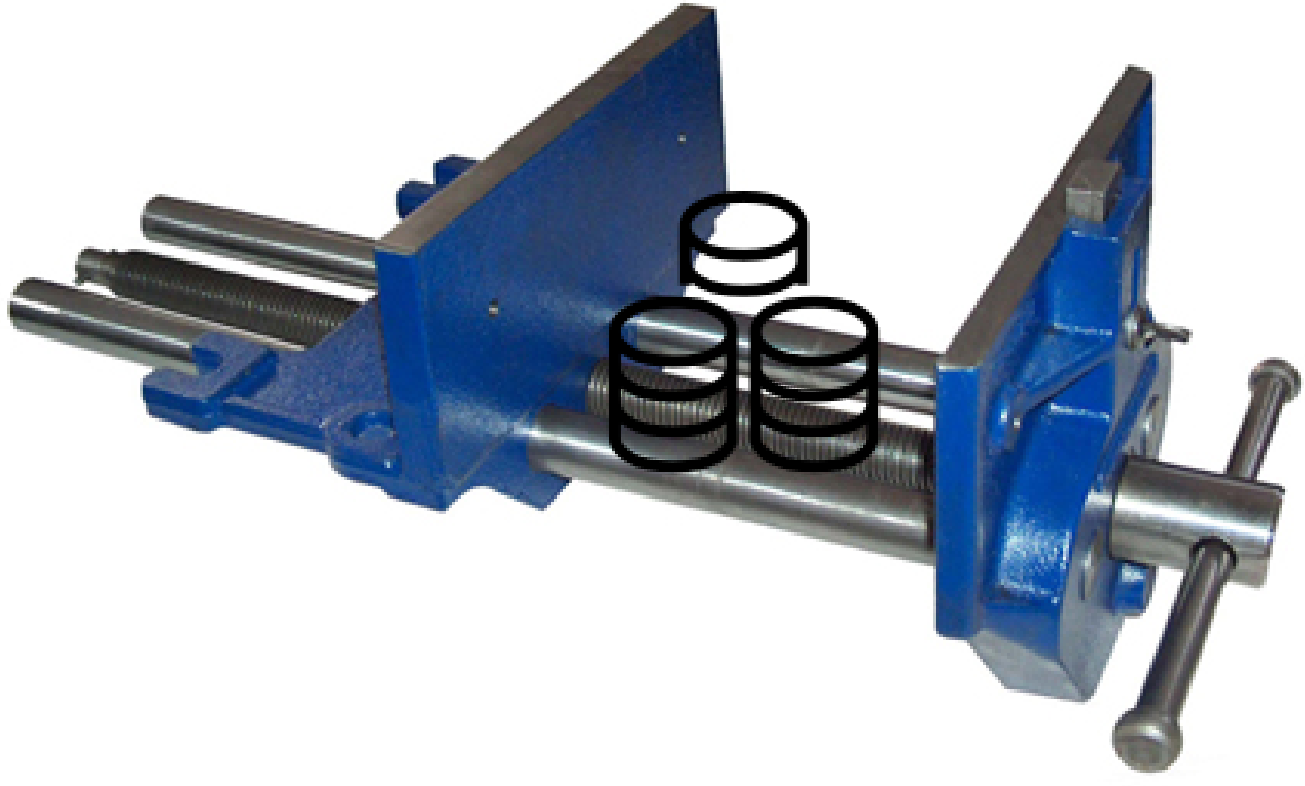
Goes out today.

Any questions about Lab 2 / Quiz 2 from last week?

Compression

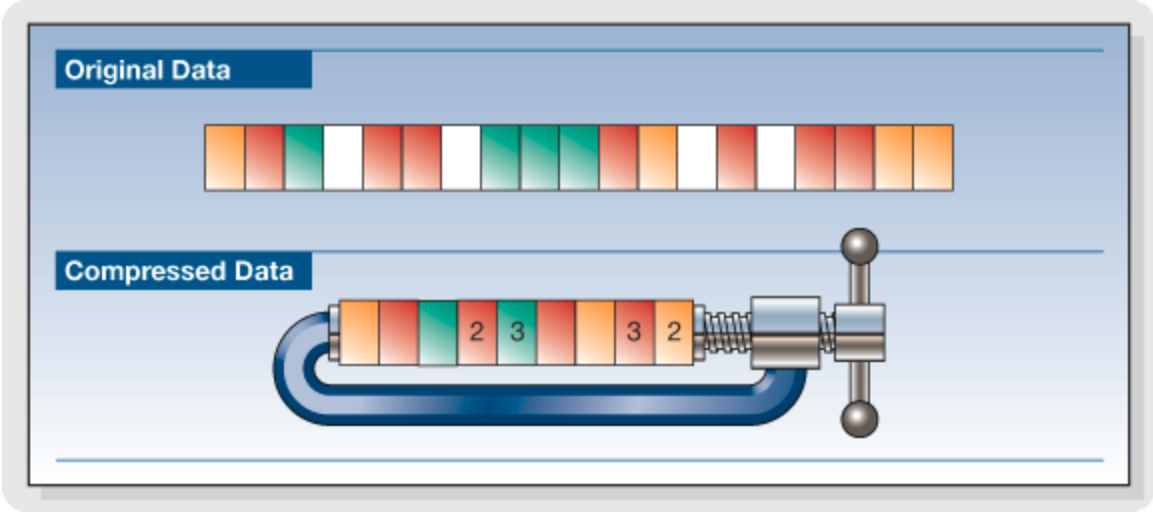


Compression



Why? Because we want to use less space!

Compression



- In this example:
  - White boxes are useless; they tell us no useful information
  - When boxes of same color are next to each other, we can just use one and say how many copies there are

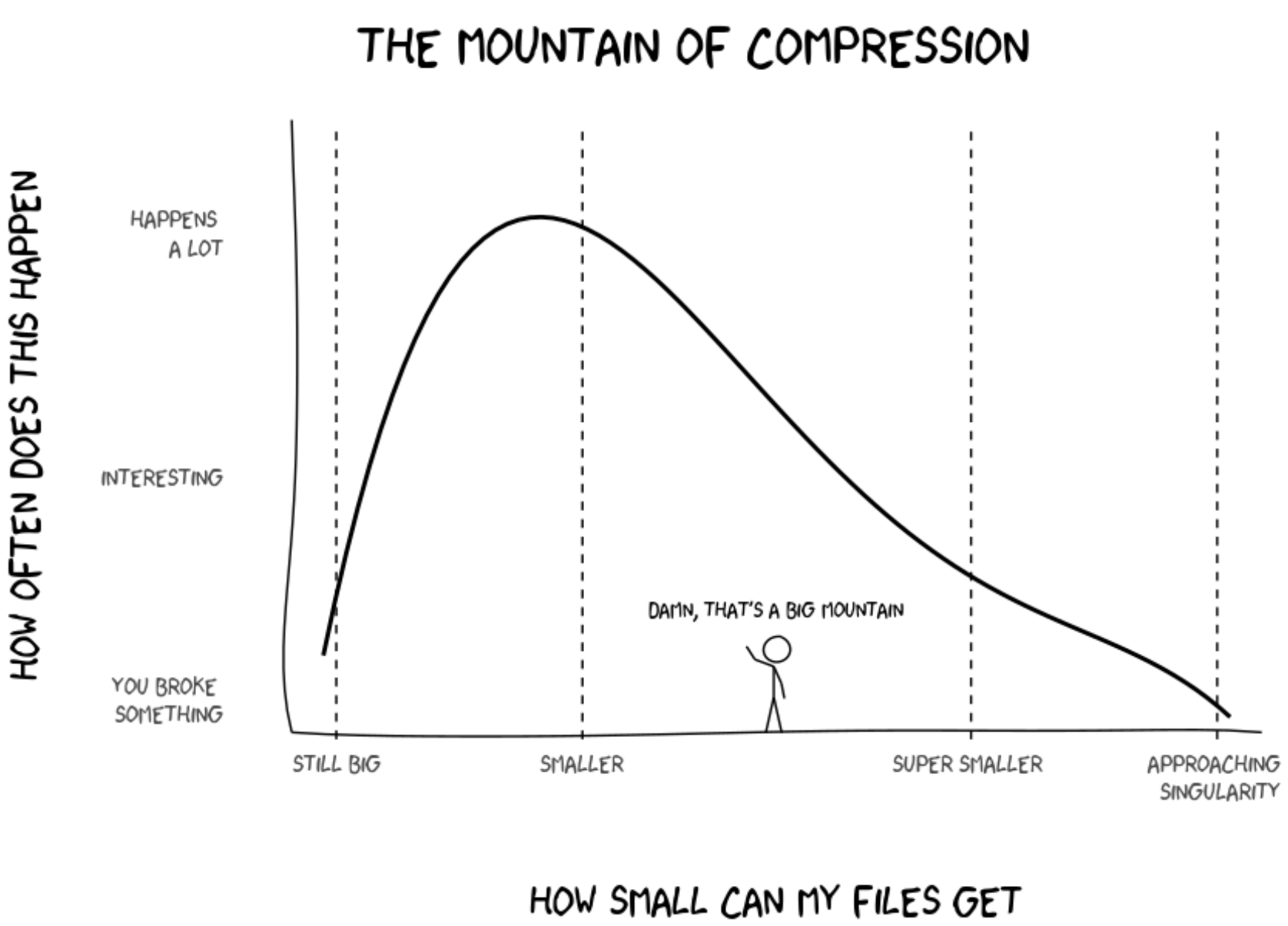
Compression

Compression isn't perfect.

A compressor may not be able to get rid of extra information, so you may end up with the same file size.

If you're looked favorably upon by the Gods, you'll get something small.

To demonstrate this, I made a plot.


THE MOUNTAIN OF COMPRESSION
HOW OFTEN DOES THIS HAPPEN
HAPPENS A LOT
INTERESTING
YOU BROKE SOMETHING
DAMN, THAT'S A BIG MOUNTAIN
STILL BIG
SMALLER
SUPER SMALLER
APPROACHING SINGULARITY
HOW SMALL CAN MY FILES GET

Compression

Lossy

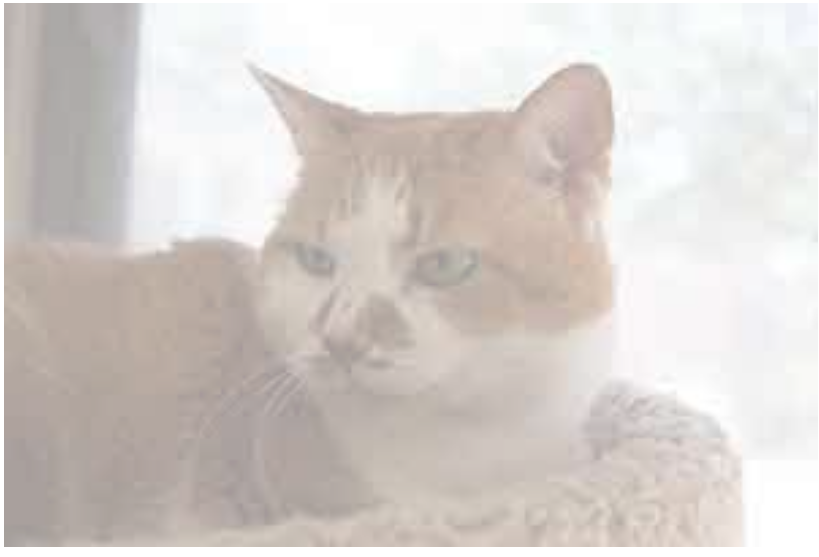
Used for some multimedia files

Lossless

Used for everything else

Compression

Lossy



Used for some multimedia files

Lossless

Used for everything else

Compression with Linux

- Compressed data on Linux called archives
  - Comes from "tape archives"
- `gzip`: Compressed files stored with `.gz` extension
  - `gzip -c file > file.gz` will make `file.gz`
  - `gzip -l file.gz` will tell you information about the compressed file
  - `gzip -[1-9]` will use compression strength, 1 through 9
  - `gzip -cd file.gz > file` will decompress to `file`

```
gzip -c vice.jpg > vice.gz
gzip -l vice.gz
         compressed        uncompressed  ratio uncompressed_name
              63721               77210  17.5% vice
```

Compression with Linux

`bzip2`: Better compression than gzip but takes longer; ext: `.bz2`

- `bzip2 -c file > file.bz2` will make `file.bz2`
- `bzip2 -[1-9]` will use compression strength, 1 through 9
- `bzip2 -cd file.bz2 > file` will decompress to `file`

Compression with Linux

- `xz`: Better compression and faster than bzip2; ext: `.xz`
  - `xz -c file > file.xz` will make `file.xz`
  - `xz -l file.xz` will tell you information about the compressed file
  - `xz -[1-9]` will use compression strength, 1 through 9
  - `xz -cd file.xz > file` will decompress to `file`

`tar` - The Standard

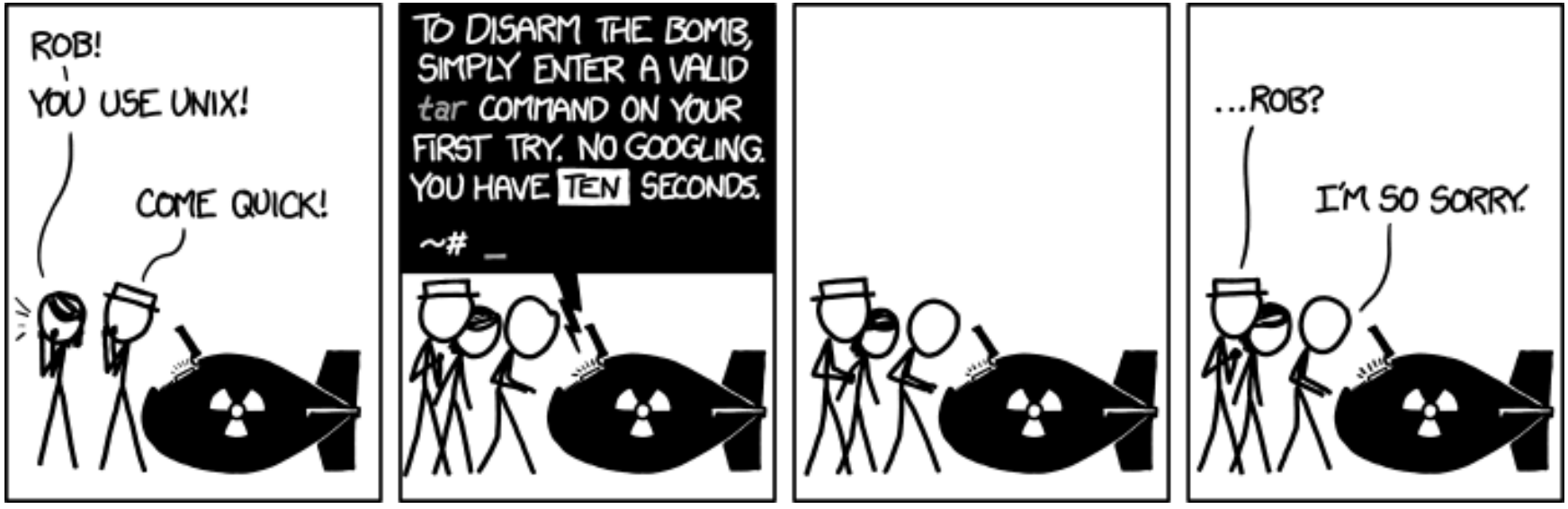
Compression with Linux: `tar`

- `tar` is a wrapper, used to first setup an archive with no compression that preserves permissions, structure, etc.
- A `.tar` file can then be compressed, making a `.tar.[]` file.

Methods

- `tar -cvf dir.tar dir` wraps, making dir into dir.tar
- `tar -xvf dir.tar` unwraps, making dir.tar into dir
- `tar -tvf dir.tar` will "peek inside" for information
- **Compression**: `-z` is gzip, `-j` is bzip2, and `-J` is xz
  - `file.tar.gz`, `file.tar.bz2`, `file.tar.xz`
  - *ex:* `tar -xzvf compressed.tar.gz` to extract

Compression with Linux: `tar`



- Since there are lots of flags, they can be easy to forget. Here is a mnemonic:
  - "xzvf": e**X**tract **Z**e **V**ucking **F**iles
  - "czvf": **C**ompress **Z**e **V**ucking **F**iles

Compression with Linux

- `zip` is commonly used for Windows, but not for Unix
- But, just in case, it is supported:
  - `zip compressed.zip file` will compress file into compressed.zip
  - `unzip compressed.zip` will extract the files

Accessing Filesystems

Operating systems need to *mount* filesystems to interact with them.

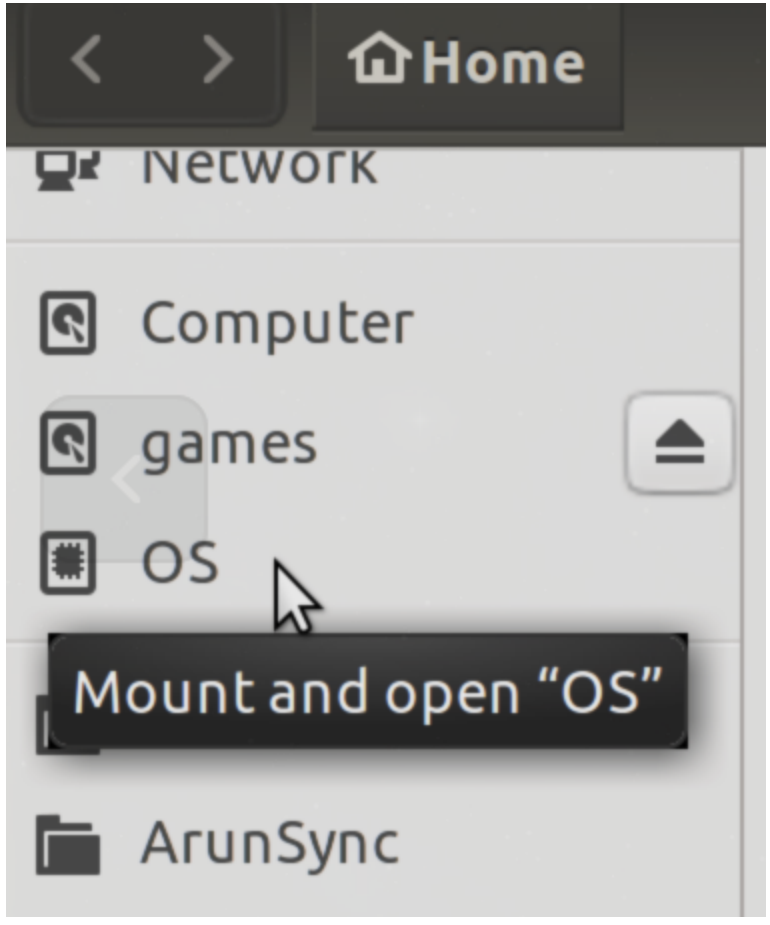
When you're done, the filesystem needs to be unmounted.

On Windows, this is done automatically (although you should click the "eject safely" button).

On Linux, we usually need to do this ourselves. Sometimes you can do this through a file manager, like with Nautilus (Ubuntu).

Mounting and Unmounting - Visually

Nautilus (file manager for Ubuntu)



Mounting and Unmounting - Shell

Your file manager automates this process, such as in Ubuntu.

Check Devices

`lsblk` will list the available partitions to mount.

Mounting

```
mount /dev/[partition] [mountpoint]
```

ex: `mount /dev/sda1 /media/adunna/GameDrive`

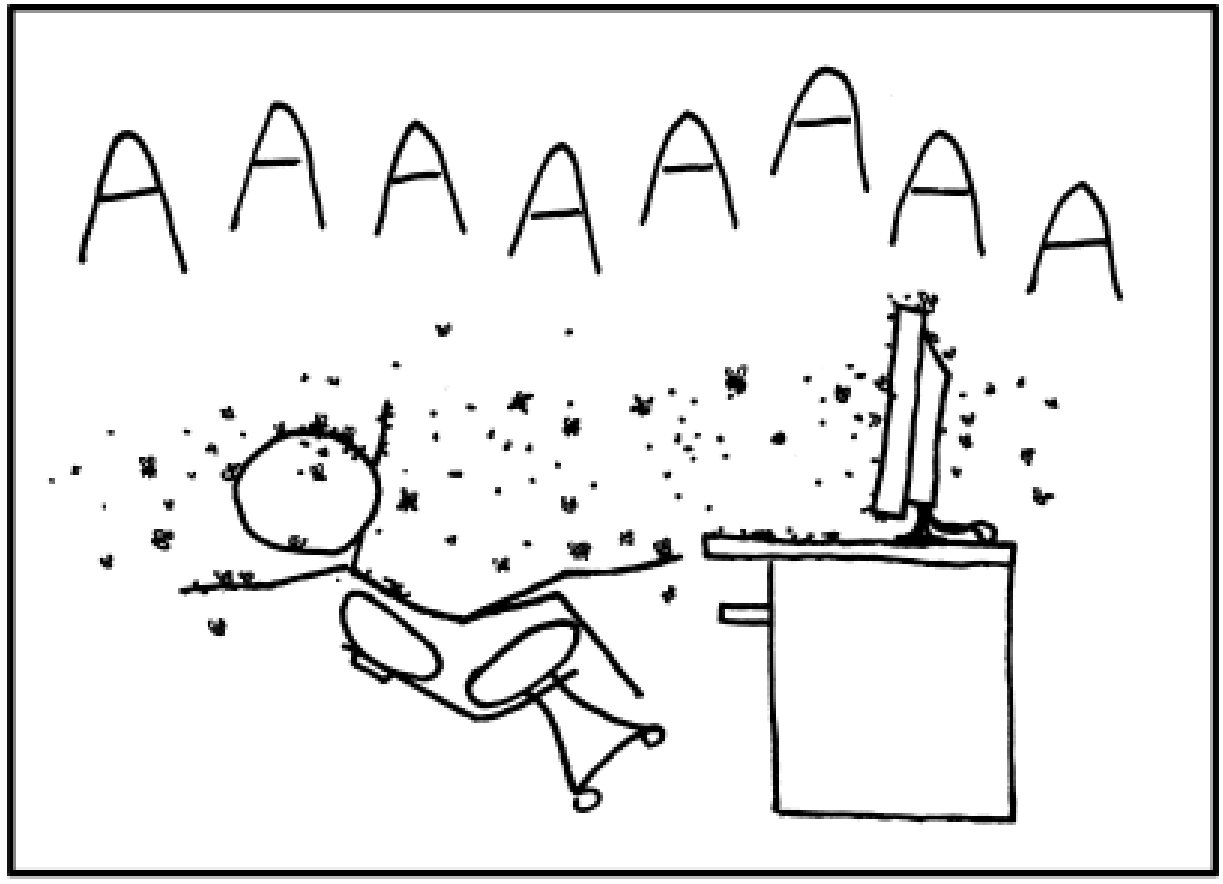
- You can mount as read-only with `-r`, which some drives need

Unmounting

```
umount [mountpoint]
```

ex: `umount /media/adunna/GameDrive`

Package Managers



MY PACKAGE MADE IT INTO DEBIAN-MAIN BECAUSE IT LOOKED INNOCUOUS ENOUGH; NO ONE NOTICED "LOCUSTS" IN THE DEPENDENCY LIST.

Package Managers

What?

A **package manager** is software that manages other software for you.

Some examples are the iOS App Store and the Google Play Store.

Likewise, many Linux distributions have their own package managers.

Why?

As software grows, dependencies become more and more complex.

So, it's easier to have a manager to take care of them for you.

Debian Family: Debian, Ubuntu, Elementary OS, Linux Mint

The Debian family has its own package manager called **dpkg**, the Debian Package Management System.

Debian packages end with the extension **.deb** so if you need to install software from a website, get the **.deb** file.

dpkg also has frontends, which you should use.

Note that for many of these commands, you will need sudo or root access.

`dpkg -i [package].deb`: Install package from file.
`dpkg -r [package]`: Removes installed package.

Package Managers: Design

Package managers centralize around repositories, which are essentially giant collections of packages.

By default, most Linux distributions have their own repositories of packages.

Sometimes, you'll have to add repositories to install third-party software.

Package managers retain local copies of the lists of packages contained in repositories, and usually have *update* commands to update these lists.

Debian Family: dpkg Frontends

apt: Advanced Packaging Tool

Usually comes default, and is the successor to **apt-get**. Many guides have **apt-get**, but you can substitute the command for **apt**.

`apt update`: Update list of packages from repositories.
`apt search [pkg]`: Search packages for specified package.
`apt install [pkg1] [pkg2] ...`: Install given package(s).
`apt remove [pkg1] [pkg2] ...`: Remove given package(s).
`apt upgrade`: Upgrade installed packages.

aptitude

You'll usually have to install this separately with `apt install aptitude`. It uses the same commands and is more extensive.

apt Setup

apt stores its package lists in `/etc/apt/`.

The default package list is `/etc/apt/sources.list`.

Extra lists are contained in `/etc/apt/sources.list.d/`, such as `/etc/apt/sources.list.d/google-chrome.list`.

Structure

```
deb [URL] [VERSION] [COMPONENTS]
```

Examples

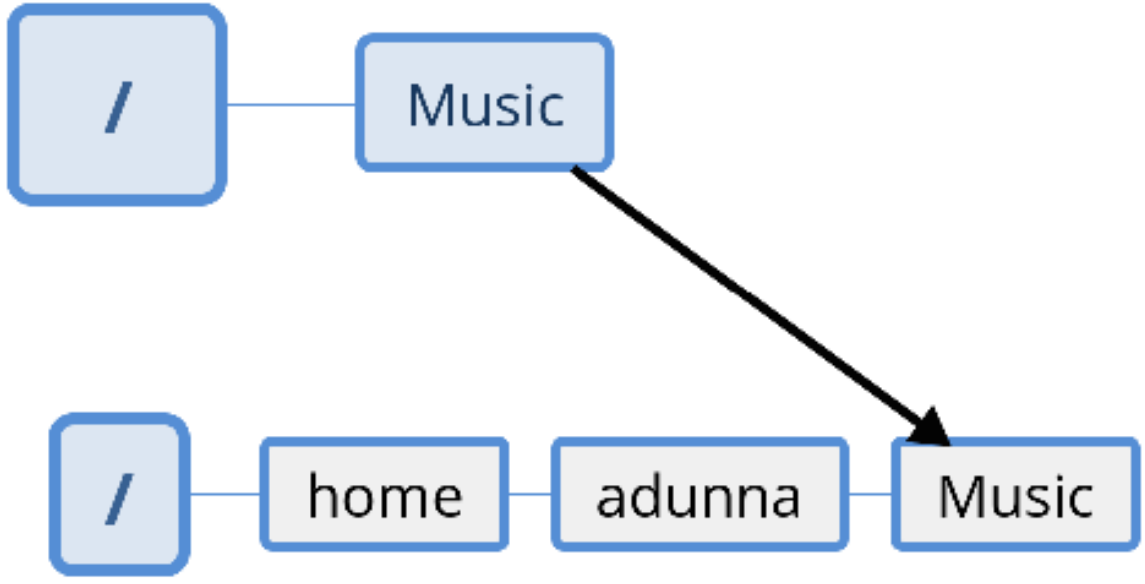
```
deb http://us.archive.ubuntu.com/ubuntu/ xenial universe

deb http://ftp.debian.org/debian squeeze main contrib

deb https://repo.skype.com/deb stable main
```

Important But Unrelated: Links

Links

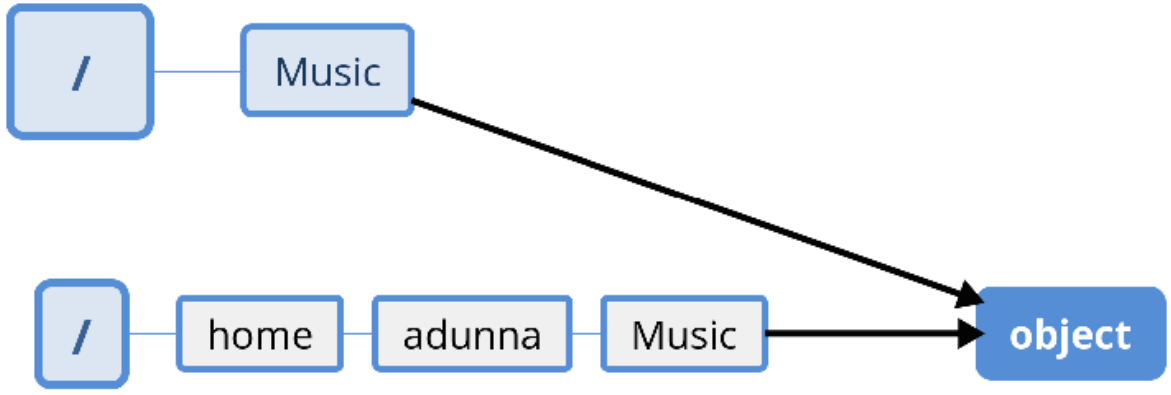
Links are like shortcuts on a file system.

That is, you can create files that point to other files or directories. These are called links.



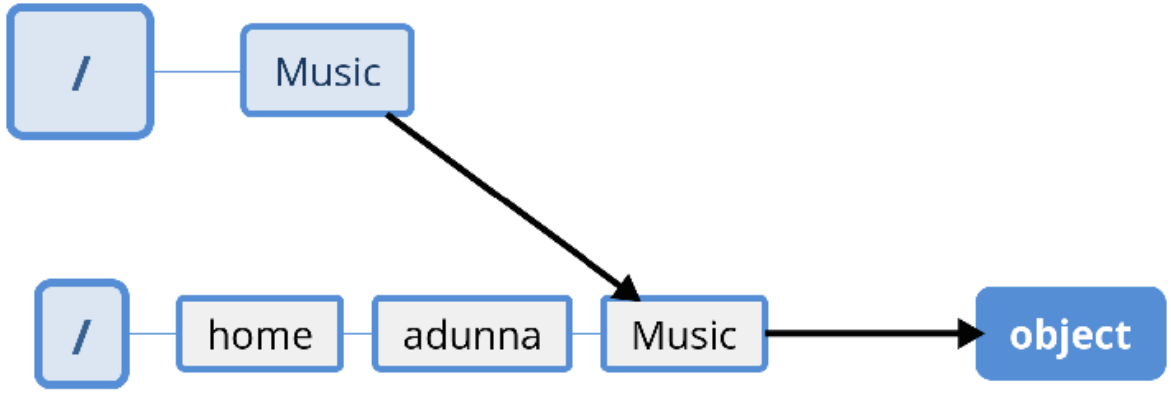
There are two types of links: hard links, and symbolic (soft) links.

Hard Links



- A file in a system points to the actual object on the disk.
  - These objects are called inodes.
- A **hard link** checks where the file points to, and then points to that inode as well.
- When you delete a file, it removes that link to the inode, but doesn't delete the data.
- When all links to the inode have been removed, the actual data is deleted.

Symbolic (Soft) Links



- A **symbolic link** points to the file that points to the data.
- When you delete the file, the actual data is deleted as well. The symbolic link will remain, but it will not work and is called a "broken link".

Hard vs. Symbolic Links

Hard Links

- References the inode (data)
- Can only link to files (not directories)
- Cannot reference files across disks (*ex:* external drive)
- Will remain usable even if the file is deleted, since the inode (actual data) will not be deleted

Symbolic Links

- References the file name and directory structure
- Can link to directories and reference files across disks
- Will exist as a broken link if the file is deleted, since the inode (actual data) will be deleted

When Should You Use _____?

1. I want to point `/Games` on my laptop's drive to an external hard drive for game storage.
2. I want to point `~/Documents/Thesis/thesis.pdf` to `~/Documents/Work/School/Thesis/thesis.pdf` and am prone to accidentally deleting files.
3. I want to point `~/Documents/Thesis/` to `~/Documents/Work/School/Thesis`.

Using Links in Linux

Hard Links

`ln [source] [link]`

ex: `ln ~/Documents/thesis.pdf ~/thesis.pdf`

And to remove the link, `rm ~/thesis.pdf`.

Symbolic Links

`ln -s [source] [link]`

ex: `ln -s ~/Documents/Thesis ~/Thesis`

You can remove symbolic links in the same way as hard links.

But **important**: Do not use `rm link/`; only use `rm link`. If you use the */*, it will go into the directory and remove files inside.

Wrap Up

Lab/Quiz 3 due **Sunday 2/17** at **11:59pm**.

Let me know if you have any questions about it, or post to Piazza.

Next Time

- Local vs. Remote
- Networking Commands
- More SSH
- Version Control