File Management

CIS 2230 Linux System Administration
Lecture 7a
Steve Ruegsegger

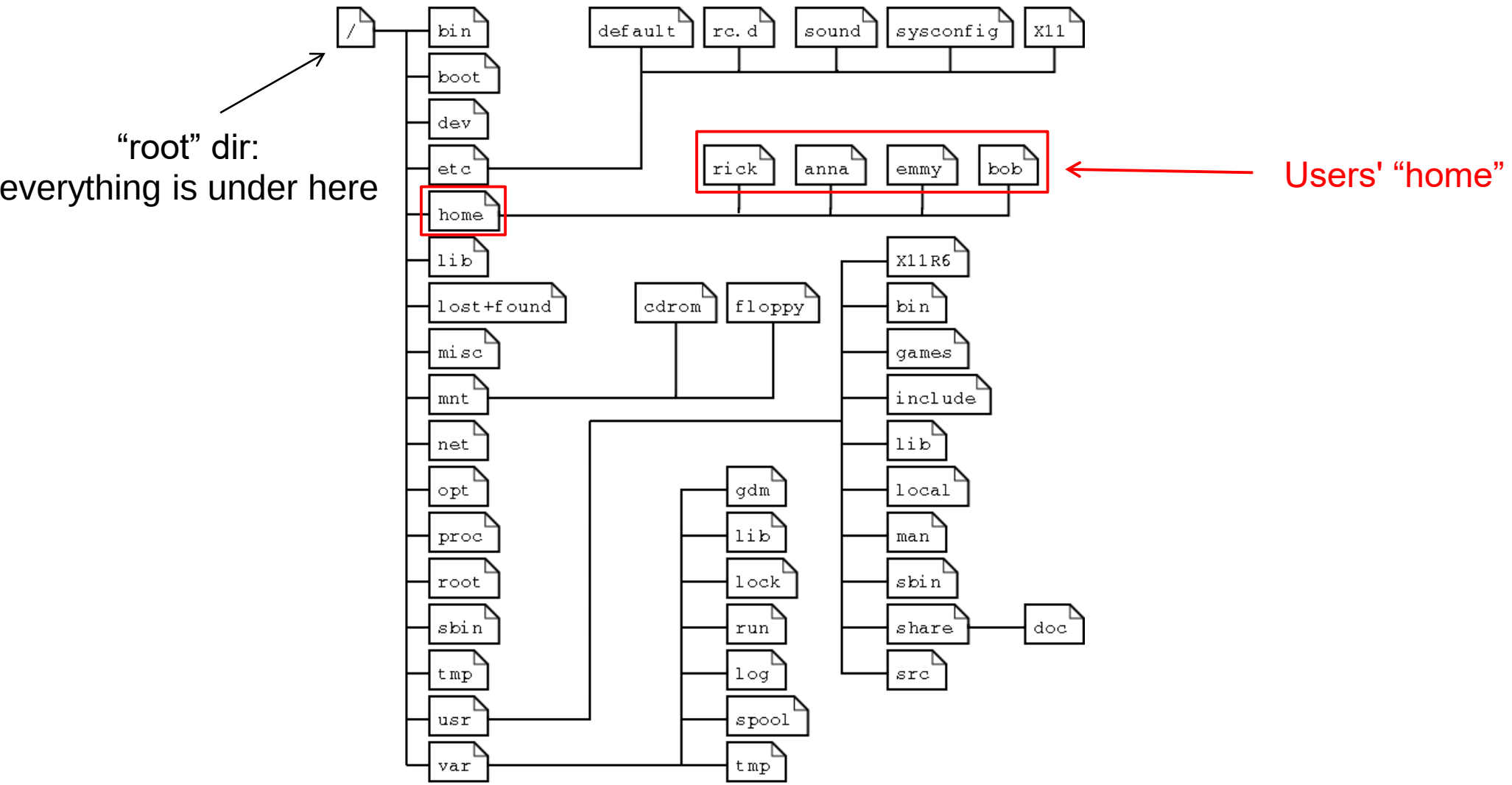
Review

- How do you define a shell variable? (e.g. set A to value 4)
- What does `$(( ))` do in a shell?
- What command prints a string to the shell?
- What are the 2 operators for "command substitution?"
- Name 2 key "pre-defined" linux env vars
- What are the commands to print all (exported) env variables?
- What is a "glob"?
- What are the 3 glob characters?
- What does `$ echo *` do and what does `$ echo "*"` do?
- What key combination finishes/completes a file or command for you?

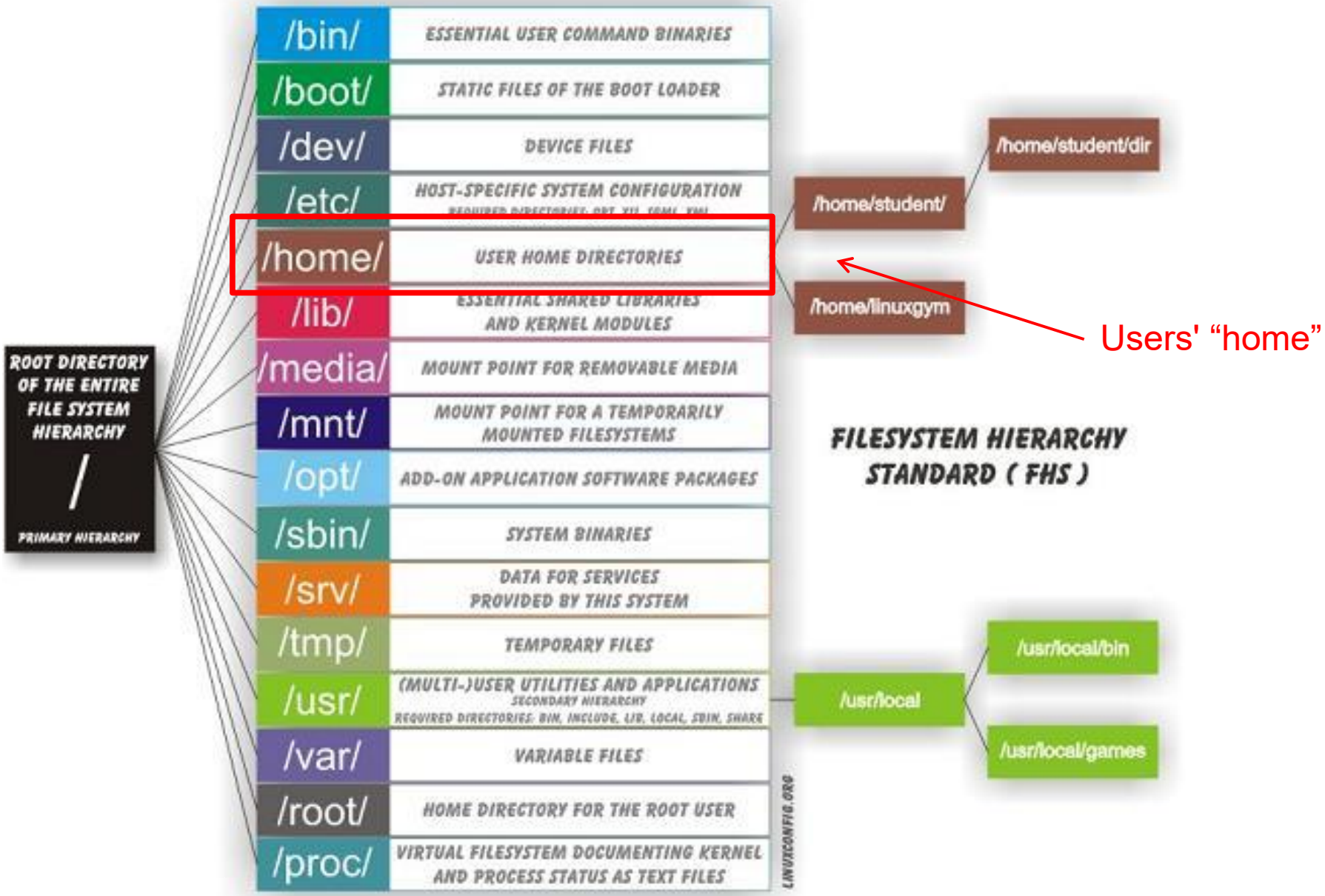
Files and Directories Overview

- Files contain data
- Directory is a collection of files and/or other directories
- Directories in directories in directories for a "hierarchy"
- The 'top level' of the hierarchy is the *root directory*
  - *(Not to be confused with the root user)*
  - /
- Files and directories can be named by a path
  - Shows programs how to find their way to the file
  - Dir names are separated by slashes (/)
  - file: `/home/steve/logs/userlog.log`
  - dir: `/usr/local/bin/`
    - Usually an extra slash at the end of a path *makes no difference*

Graphical example of linux directory tree structure



FHS - filesystem hierarchy standard



Key file commands

4 – Manipulating Files And Directories

At this point, we are ready for some real work! This chapter will introduce the following commands:

- `cp` – Copy files and directories
- `mv` – Move/rename files and directories
- `mkdir` – Create directories
- `rm` – Remove files and directories
- `ln` – Create hard and symbolic links

These five commands are among the most frequently used Linux commands. They are used for manipulating both files and directories.

TLCL

Working directory

- At all times, your shell has a “current working directory” (cwd)
  - `$ pwd  # print working directory`
  - Unless otherwise specified, *everything* 'happens' in the cwd
  - We say you are “*in*” that directory
- `$ ls` lists files in the cwd
- `$ cat userlog.log` looks only in the cwd for the file `userlog.log`

```
$ cd
$ cd labs/lab04
$ cat lab4.txt
```

Special Dot Directories

- *Every* directory contains two (2) special filenames which help making relative paths:
  - The directory ".." points "up" to the parent directory
  - The directory "." is the current directory
  - Since each dir has a ".." pointer, then they can be strung together to go up multiple levels.
- Example:
  - If cwd is "`/home/steve/Accounts`"
  - then "`..`" points to _____
  - And "`../..`" points to _____
- Commands:

```
$ ls -a
$ ls ..
$ cd ../..
```

Absolute vs Relative Paths

- An absolute path starts at the root of the directory hierarchy, and names all directories under it:

  `/etc/hostname`

  `/home/steve/Documents`

  - It *absolutely* must start with "/" (root)

- A relative path starts with the current directory, wherever that may be!
  - e.g. absolute: `$ cat /home/steve/logs/userlog.log`
  - From `/home`: `$ cat steve/logs/userlog.log`
  - From `/home/steve`: `$ cat logs/userlog.log`
  - The file's path is *relative* to your current directory

Absolute vs Relative Paths in cd

- When you `cd`, you can use either type of path.
  - Sometimes, one makes more sense and is more 'stable' in your programming
- For example, the following sets of directory changes both end up in the same directory:

```
$ cd /usr/share/doc
```

or ...

```
$ cd /
$ cd usr
$ cd share/doc
```

- Often, relative paths have ".." (up-dir)

```
$ ../../src/project/runme.pl
```

Making and Deleting Directories

- Make a new, empty, directory: `mkdir`
  - Relative: `$ mkdir Accounts`
  - Absolute: `$ mkdir /home/steve/personal/Accounts`
- Delete an empty directory, use `rmdir`:

```
$ rmdir OldAccounts
```

- If not an empty dir: use rm with the -r (recursive) option

```
$ rm -r OldAccounts
```

- Be careful: `$ rm -r` can be a dangerous tool if not careful

Paths to Home

- 4 shortcuts to 'home' (usually `/home/<userid>`)
  1. `$ cd`
  2. `$ cd $HOME`
  3. `$ cd ~`
  4. `$ cd ~steve`
- Notes:
  - The default for `cd` is the user's home
  - `$HOME` is an env variable for the user's home
  - The symbol `~` (tilde) is an abbreviation for your home directory
  - `~<user>` is short for that user's home
  - The `~` is expanded *by the shell*, so programs only see the completed path
- You can also use `~` in command args:
- `$ cat ~alice/notes.txt`

Files

File Extensions

- It's common to put an extension, on the end of a filename
- The extension can indicate the type of the file:

```
.txt Text file
.gif Graphics Interchange Format image
.jpg Joint Photographic Experts Group image
.mp3 MPEG-2 Layer 3 audio
.gz  Compressed file
.tar Unix 'tape archive' file
.tar.gz, .tgz Compressed archive file
```

- On Unix and Linux, file extensions are just a convention
  - The kernel just treats them as a normal part of the name
  - A few programs use extensions to determine the type of a file
- Summary: extensions are for the user, they mean nothing for Linux

It's magic...

```
FILE(1)                          BSD General Commands Manual                          FILE(1)

NAME
     file – determine file type

SYNOPSIS
     file [-bchiklLNnprsvz0] [--apple] [--mime-encoding] [--mime-type] [-e testname] [-F separator]
          [-f namefile] [-m magicfiles] file ...
     file -C [-m magicfiles]
     file [--help]
```

```
$ file viva_la_vida.mp3
viva_la_vida.mp3: Audio file with ID3 version 2.4.0, contains:
MPEG ADTS, layer III, v1,  48 kbps, 44.1 kHz, JntStereo
```

Hint: rename the extension of a filename, and re-try `$ file`. Can you trick it?

Hidden Files

- Simple rule: files/dirs which start with . are considered 'hidden'
- 'Hidden' simply means not listed by 'ls' by default
  - You can still read & write them.
- You can ask ls to display all files with the -a (all) option:

```
$ ls -a
. .. .bashrc .profile report.doc
```

- Hidden files and dirs are often used for configuration files
  - Usually found in a user's home directory

Copy (review)

```
$ man cp
```

```
CP(1)                                   User Commands                                   CP(1)

NAME
       cp - copy files and directories

SYNOPSIS
1      cp [OPTION]... [-T] SOURCE DEST
2      cp [OPTION]... SOURCE... DIRECTORY
3      cp [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
       Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
```

- 3 ways to use cp:

1. File to file (implicit or explicit)

```
$ cp ~/logs/userlog.log /mnt/bkup/userlog.log.`date +%F:%T`
```

2. file(s) to directory (last arg is a dir)

```
$ cp *.txt *.doc ../logs
```

3. Also file(s) to directory, but use -t <dir> option

```
$ cp -t ../logs *.txt *.doc
```

Move/rename

- "moving" is how you "rename" in linux
- Same syntax for moving files

```
MV(1)                                  User Commands                                  MV(1)

NAME
       mv - move (rename) files

SYNOPSIS
       mv [OPTION]... [-T] SOURCE DEST
       mv [OPTION]... SOURCE... DIRECTORY
       mv [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
       Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.
```

- Also 3 ways to use, just like cp.
- Note: I can also use move to rename!

  1) Rename in cwd: `$ mv file1.txt file2.txt`

  2) Move and rename: `$ mv file1.txt ../file2.bak`

Creating a file with another date

- This might be useful some day:

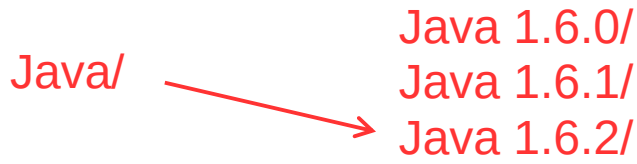
```
$ date
Sun Aug 12 22:20:34 EDT 2012

$ touch -d '2012-01-01' foo.bar

$ ls -l
total 0
-rw-rw-r-- 1 steve steve 0 Jan  1  2012 foo.bar
```

compression -- gzip

- As Sys. Admins, we get lots of text files, which compress very well
- `gzip` is the standard compression from GNU
  - open source, free, copyleft, etc.
  - Other unix use older "`compress`" command
- extension:
  - a gzip file ends in `.gz`
  - a compress file end in `.Z`
- multiple files:
  - gzip is a “per-file” compression
  - if you want more than 1 file (like Windows zip), there is a linux version of "zip", but usually files are first tar-ed then compressed into a single tar.gz
- commands:
  - `$ gzip <files>` - to zip files & add .gz ending
  - `$ gunzip <files>` - to unzip & remove .gz ending (same as gzip -d)

Symbolic File Links

- We'll just touch the surface here
- Another file *type* is a symbolic link
- A symlink is a pointer to some other file or directory
- When you request access to the symlink, the kernel recognizes the link and 'forwards' access to the pointed-to file.
  - File A ⟶ File B
- Symlinks allow you to keep a file (or directory) in one place, but 'pretend' it lives in another
  - For example, to ensure that an obsolete name continues to work for older software



- 2 types: hard link and symbolic (or soft) link

Creating Symbolic Links

- A symlink is created with the ln -s command
- Its syntax is similar to cp — the 'source' or 'real file' comes first, then the link name you want to create:

```
$ ln -s real-file file-link
$ ln -s real-dir dir-link
```

- The file looks 'normal'. Use `$ ls -l` to see that it's a link.

```
$ ls -l
lrwxrwxrwx 1 bob bob 9 Jan 11 15:22 file-link -> real-file
lrwxrwxrwx 1 bob bob 8 Jan 11 15:22 dir-link -> real-dir
```

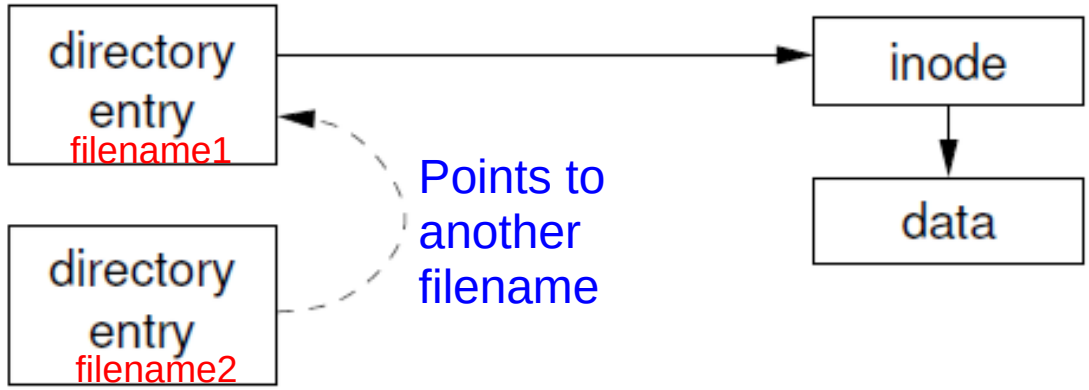
- I still like... `alias ls='ls -F'`

Hard Links

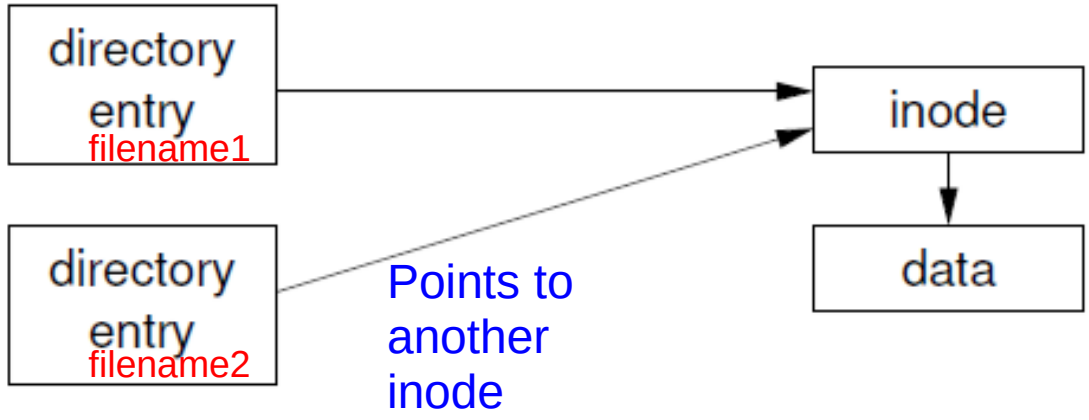
- A **hard link** associates two or more filenames with the *same* inode
- It does not point to the filename of the original file. It points to the data along with the original filename.
- More detail is for the Adv. Class in Filesystems lecture
- For now, the “new” hard link has just as much 'claim' on the data as the original filename.

Hard-link & symlink

- A symbolic link refers to filename, which in turn refers to an inode:



- A hard link is a normal directory entry, referring directly to an inode:



This is why the inode does not contain the filename!
Because it's data can have more than 1 filename!

What's the difference?

- Hard links
  - Cannot link directories
  - Cannot cross file system boundaries (b/c inums don't match!)
- symlinks
  - Can create links between directories
  - Can cross file system boundaries
- What if the source of the link is moved or removed?
  - Symbolic links are not updated.
    If the source is removed, the *symlink doesn't know it*!
  - Hard links always refer to the source – the inum itself!

tarballs

- `tar` stands for tape archive
- The command not used as much for actual tape archiving, but for stringing together multiple files and dirs into one file.
  - Note that (Windows) "zip" does compression and grouping
  - But "gzip" only does compression
  - Therefore, tar does the grouping into 1 file
- 3 basic commands/functions: (`$ man tar`)

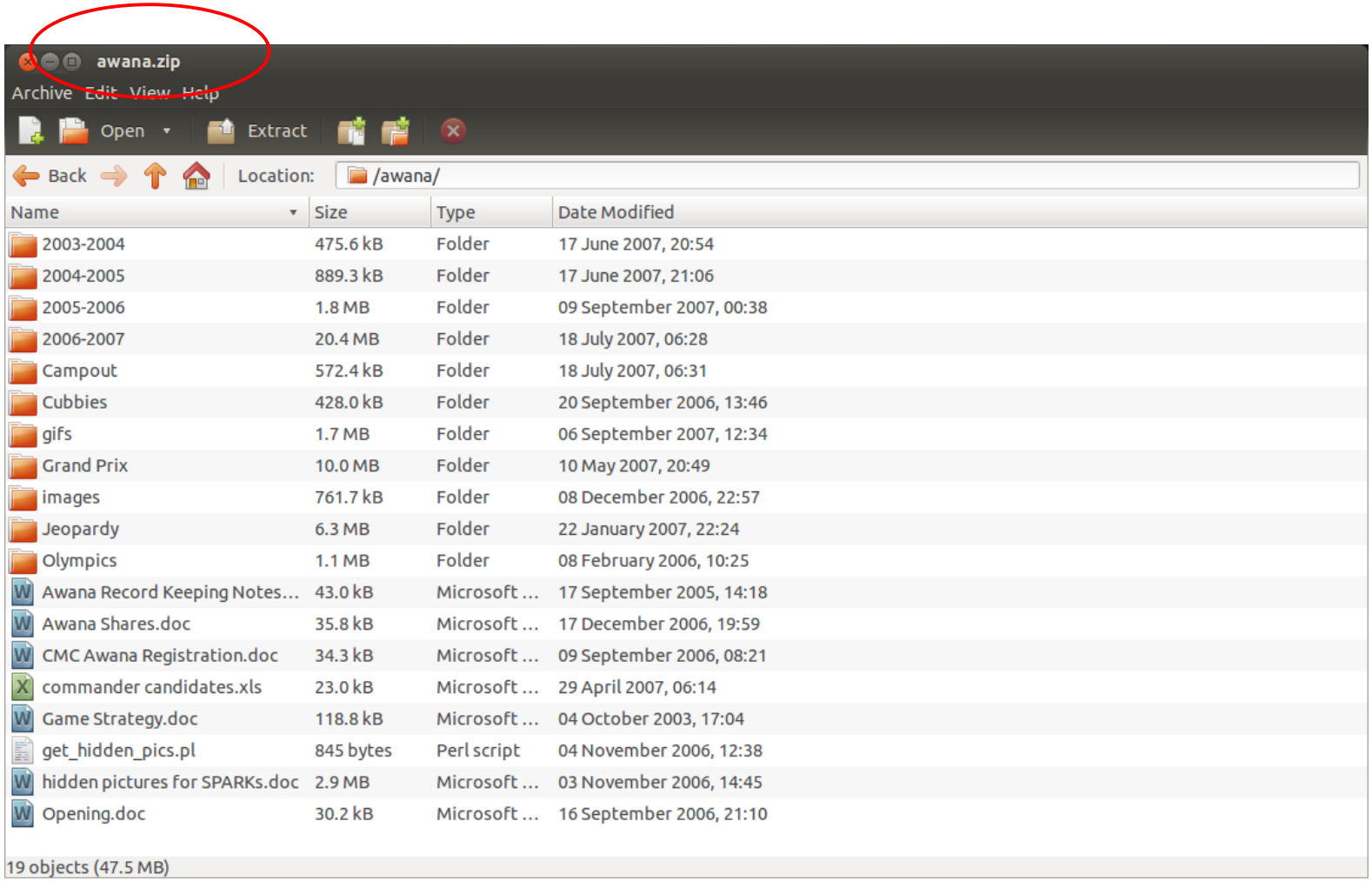
```
tar <function> <file...>

tar xvf <tarball>
tar cvf <files...>
tar tvf <tarball>
```

memorize

- Extraction (x) occurs in the cwd (unless you use an option to redefine)
- Create (c) keeps is recursive, keeps the dir structure, but removes leading "/"
- That tar file can be compressed with gzip (.gz) or compress (.Z)
  - `file.tar.gz, file.tar.Z, file.tgz`

GUI – file-roller

- Of course, there is a GUI for .zip, tar.gz, etc.
- `$ file-roller <file>`



Name	Size	Type	Date Modified
2003-2004	475.6 kB	Folder	17 June 2007, 20:54
2004-2005	889.3 kB	Folder	17 June 2007, 21:06
2005-2006	1.8 MB	Folder	09 September 2007, 00:38
2006-2007	20.4 MB	Folder	18 July 2007, 06:28
Campout	572.4 kB	Folder	18 July 2007, 06:31
Cubbies	428.0 kB	Folder	20 September 2006, 13:46
gifs	1.7 MB	Folder	06 September 2007, 12:34
Grand Prix	10.0 MB	Folder	10 May 2007, 20:49
images	761.7 kB	Folder	08 December 2006, 22:57
Jeopardy	6.3 MB	Folder	22 January 2007, 22:24
Olympics	1.1 MB	Folder	08 February 2006, 10:25
Awana Record Keeping Notes...	43.0 kB	Microsoft ...	17 September 2005, 14:18
Awana Shares.doc	35.8 kB	Microsoft ...	17 December 2006, 19:59
CMC Awana Registration.doc	34.3 kB	Microsoft ...	09 September 2006, 08:21
commander candidates.xls	23.0 kB	Microsoft ...	29 April 2007, 06:14
Game Strategy.doc	118.8 kB	Microsoft ...	04 October 2003, 17:04
get_hidden_pics.pl	845 bytes	Perl script	04 November 2006, 12:38
hidden pictures for SPARKs.doc	2.9 MB	Microsoft ...	03 November 2006, 14:45
Opening.doc	30.2 kB	Microsoft ...	16 September 2006, 21:10

19 objects (47.5 MB)