tar: tape archive

- Utility to manage archives
- Can preserve directories, links, owners, groups, permissions
- Usage is a little tricky.
- Mandatory main switches (must specify exactly one):
  - `c` : create new archive
  - `r` : append to existing archive
  - `t` : list contents
  - `x` : extract files
  - `...` : (there are others)
- An extremely useful optional switch (check your `man` pages):
  - `f` : specify a file instead of the tape drive
    - use "-" for standard input/output
    - Tar archive files are called tarballs

Simple tar examples

- Create an archive stored in file `foo.tar`, that contains a copy of directories `Project1`, `Project2`

```
tar cf foo.tar Project1 Project2
```

- List contents of archive `foo.tar`

```
tar tf foo.tar
```

- Extract (copies of) files from archive `foo.tar`

```
tar xf foo.tar
```

The archive is not modified when you do this

sort: sort a file

- Usage: `sort [file] [file]`
  - Concatenates files and sorts them
  - No file specified: read from standard input
- Lots of useful options, check your `man` pages
  - `-k` : key position (default: 1) (column to sort by)
  - `-n` : numeric sort (otherwise – alphabetical sort)
  - `-r` : reverse sort
  - `-u` : merge unique

```
prompt$ sort catfile.txt
An editor would be better.
But it is better than using echo.
I'm making this file
with cat and redirection.
prompt$ sort -k 2 catfile.txt
with cat and redirection.
An editor would be better.
But it is better than using echo.
I'm making this file
prompt$
```

How to rename all ".cc" files as ".c"?

```
for f in *.cc; do
  bn=$(basename -s .cc $f)
  mv $bn.cc $bn.c
done
```

- Usage: `yes [expletive]`
- Writes `expletive` to standard output, forever
- If no expletive is given, default is "y"

```
prompt$ yes no
```

grep: select lines that match a pattern

- Usage: `grep pattern [file] [file] ...`
- No files specified: reads from standard input
  - Do you see a pattern yet?
- Simple pattern: plain text
- Fancier patterns: later in the semester
- Writes lines that contain the pattern

```
prompt$ ls
1.SpeakToMe.mp3    4.TheGreatGig.mp3  7.AnyColourYo.mp3
2.OnTheRun.mp3     5.Money.mp3        8.BrainDamage.mp3
3.Time.mp3         6.UsAndThem.mp3    9.Eclipse.mp3
prompt$ ls | grep The
2.OnTheRun.mp3
4.TheGreatGig.mp3
6.UsAndThem.mp3
prompt$
```

wc: count words, lines of a file

- Usage: `wc [file] [file] ...`
- Display the number of lines, words, and characters in each file
- If more than one file, also display the total
- If no files specified — read from standard input
- Can use switches to get just one value (e.g., just # lines)
  - "Consult your `man` pages" as usual

```
prompt$ wc catfile.txt
      4      20     108 catfile.txt
prompt$ ls | wc
     22      22     215
```

- Suppose your shell is set up so that `rm` prompts before removing each file
  - In a few lectures, we will see how to set this up
- To remove directory `Foo` and all its files:

```
prompt$ yes | rm -R Foo
```

gzip: compress a file

- Usage: `gzip [file] [file]`
- Compresses each file, adds ".gz" to file name
- If no files specified:
  - Reads from standard input
  - Writes to standard output

```
prompt$ ls -l | grep file
-rw------- 1 alice staff 62976 Jul 24 15:58 file.c
prompt$ gzip file.c
prompt$ ls -l | grep file
-rw------- 1 alice staff 14458 Jul 24 15:58 file.c.gz
prompt$
```

gunzip: uncompress a file

- Usage: `gunzip [file.gz] [file.gz]`
- Uncompresses each file, removes ".gz" from file name
- If no files specified:
  - Reads from standard input
  - Writes to standard output

```
prompt$ gunzip file.c.g.gz
prompt$ mv file.c.g file.c.gz
prompt$ gunzip file.c.gz
prompt$ ls -l | grep file
-rw------- 1 alice staff 62976 Jul 24 15:58 file.c
prompt$
```

Here is the basic syntax for creating a symlink to a file in your terminal.

```
ln -s existing_source_file optional_symbolic_link
```

You use the `ln` command to create the links for the files and the `-s` option to specify that this will be a symbolic link. If you omit the `-s` option, then a hard link will be created instead.

Install package from source (tar):

1. Unpacking: tar xf coolthing-2.0.4.tar.gz
   a. cd coolthing-2.0.4.tar.gz
2. configuring: ./configure
3. building: make
4. installing: Sudo make install

Install kernel from source (tar):

1. Lynx [The Linux Kernel Archives]
   a. Or other source
2. Config: make config; make oldconfig; make menuconfig; make x config; make localmodconfig;
3. Build: make (not sudo)
4. Install: make install; make modules_install

Creating symbolic link: Ln -s /path/from /new/path/$i/paths

8 0 / 1 point

Suppose there is a Linux utility, lsinfo, that lists various information about the system. Give a single shell command (pipeline) to display lines 7 through 13 (inclusive) of the output of lsinfo.

user$ y

Correct Answer: lsinfo | head -n 13 | tail -n 7, lsinfo | tail -n +7 | head -n 7

Question 4

Suppose you are installing a proprietary (closed-source) browser named shiny, and have already downloaded its executable and collection of plug-ins. You have already copied the executable to /usr/local/bin. The plug-ins are supposed to be installed to directory /usr/local/shiny_plugins. The authors of this software have not followed good software engineering practice, because the plugin directory location cannot be reconfigured.
Now, suppose the plug-ins are currently located in directory /shiny/plugins, and require 1.1 Gb of disk space. However, before copying them to their required location (as root) you realize that you have only 200 Mb of available space in your /usr partition. Give a specific command, or sequence of commands, that can be executed as root to complete the installation, without breaking any other installed applications. In particular, you **may not** repartition the drive or remove any files.

My solution:

```
ln -s /shiny/plugins /usr/local/shiny_plugins
```

2 1 / 1 point

This is a bogus question because Canvas won't let me just type plain text without a question attached.

True

False

Question 11

Write a bash script named Unpack that takes as arguments a list of gzipped tarballs, for example, ./Unpack first.tar.gz second.tar.gz third.tar.gz. For each argument, the script should unpack the tarball into the current working directory.

My solution:

```
#!/bin/bash
# Note there is no error checking whatsoever here
for arg; do
  tar xf $arg
done
```

3 1 / 1 point

Another bogus question to make Canvas happy.

True

False

- `cd` : Change working directory.
- `info` : Fancy browsing of the online manual.
- `ls` : List the contents of a directory.
- `man` : Browse the online manual.
- `mkdir` : Create a directory.
- `pwd` : Print working directory.
- `rmdir` : Remove a directory.
- `type` : Is a command built-in, or not?

- `cat` : Concatenate a file (to the display).
- `chgrp` : Change file group.
- `chmod` : Change permissions.
- `chown` : Change file owner.
- `cp` : Copy files or directories.
- `hexdump` : Show hex contents of a file.
- `mv` : Move files or directories.
- `reset` : Reset a trashed terminal.
- `rm` : Remove files or directories.

- `date` : Display the current date and time
- `exit` : Exit a shell
- `kill` : Signal a process
- `nice` : Run with lower priority
- `ps` : List processes
- `renice` : Adjust priority of a process
- `systemctl` : Manage services (using `systemd`)
- `wait` : Wait for one or more processes

In the shell, we can change the file descriptors around:

`command args < file` : `stdin` reads from file

`command args > file` : `stdout` writes to file

```
prompt$ cat name.txt
Bob Roberts
Let's suppose some other stuff is here
prompt$ ./hello < name.txt
What is your name?
Hello, Bob Roberts!
prompt$ ./hello > out.txt
What is your name?
Doctor Robert
prompt$ cat out.txt
Hello, Doctor Robert!
prompt$
```

- `bg` : Run a job in background mode
- `echo` : Display arguments
- `fg` : Run a job in foreground mode
- `jobs` : Display jobs
- `kill` : Signal a job
- `su` : Substitute user
- `wait` : Wait for one or more jobs
- `whoami` : Who am I

- `cat -n` : show files and number lines
- `grep` : select lines matching text
- `head` : select beginning of a file
- `less` : modern pager
- `more` : classic pager
- `sort` : sort a file
- `tail` : select end of a file
- `tee` : pipeline splitter
- `tr` : translate characters
- `wc` : count lines, words
- `xargs` : extract arguments
- `yes` : print "y" or other text, forever

Generic steps to build a kernel

Almost identical to the "generic steps to build from source code"

1. Obtain the source code
2. Read documentation
3. Configure
4. Build
5. Install
6. Test
7. Enjoy

chmod 640 foo.txt

- 6 : 4 + 2 + 0 means `rw-` for user
- 4 : 4 + 0 + 0 means `r--` for group
- 0 : 0 + 0 + 0 means `---` for other

chmod 755 public/

- 7 : 4 + 2 + 1 means `rwx` for user
- 5 : 4 + 0 + 1 means `r-x` for group
- 5 : 4 + 0 + 1 means `r-x` for other

mkdir: make (empty) directories

- Arguments: (absolute or relative) pathnames

`-p` : will create intermediate directories if necessary

rmdir: remove empty directories

- Arguments: (absolute or relative) pathnames
- Will fail if the directory is not empty

`-p` : will remove parent directories also

Adding to a file

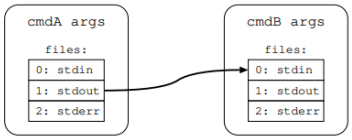
- It is possible to append to a file
  - `>> file` : `stdout` appends to file
  - `1>> file` : `stdout` appends to file
  - `2>> file` : `stderr` appends to file
- If file does not exist already:
  - Depends on shell and settings
  - May complain
  - May create an empty file (act like >)

```
prompt$ echo "And now for an important message." > out.txt
prompt$ ./hello < name.txt >> out.txt 2> /dev/null
prompt$ cat out.txt
And now for an important message.
Hello, Bob Roberts!
prompt$
```

- In a shell, a pipe sends the output of one command *directly* as input to another command
- To set this up:

```
prompt$ cmdA args | cmdB args
```

- The shell uses two processes for this



chown: change owner and group of files

Usage:

- `chown owner file1 ... filen`
  Change the owner of the specified files
- `chown owner:group file1 ... filen`
  Change the owner and group of the specified files
- `chown :group file1 ... filen`
  Change the group of the specified files

chgrp: change group of files

Usage:

- `chgrp group file1 ... filen`
  Change the group of the specified files

General utilities, useful for installing from source

- `diff` : show file differences
- `make` : build something
- `patch` : apply changes to a file
- `rsync` : remote file copy
- `tar` : manage archive files
- `unzip` : unpack a "zip" archive
- `zip` : pack a "zip" archive

Useful `ls` switches

- `-a` : show all files
  - Filenames starting with `.` are normally hidden
- `-l` : long listing
  - Default is "short listing": ordered names in columns
- `-F` : extra character displayed for the file type
  - `*` : executables
  - `/` : subdirectory
  - `@` : symlink (will discuss later)
  - `=` : socket (discussed in ComS 352)
- `--color` : Like "-F" but uses color (Linux only)
- `-G` : Like "-F" but uses color (Mac OS only)

rm: remove (delete) files

- Files to remove are passed as arguments

`-i` : interactive
- Ask before removing

`-r` : recursive

`-R` : recursive
- Will recursively remove files in subdirectories
- Be careful with this

What about stderr?

A couple of options, depending on what you want:

- `0< file` : `stdin` reads from file (same as <)
- `1> file` : `stdout` writes to file (same as >)
- `2> file` : `stderr` writes to file

Another option:

- `2>&1` : send `stderr` to where `stdout` is currently going (order is important here)

```
prompt$ ./hello 2> out.txt
Bob again
Hello, Bob again!
prompt$ cat out.txt
What is your name?
prompt$
```

Bash Scripting example:

```
#!/bin/bash

For I in example/folder/*; do

        Function 1

        Function 2

done
```

High-level package management

- `apt` : Debian's package system
- `dnf` : Replacement for yum
- `yum` : Fedora/Red Hat package system

Low-level package management

- `dpkg` : Debian Package; for `.deb` files
- `rpm` : Red Had Package Manager; for `.rpm` files

Suppose you want to install an open-source package named libcool. Give the specific command to execute (as root) to install this package using yum.

root$ yum libcool

Correct Answer: **yum install libcool**

0 / 1 point

Suppose you want to install an open-source package named libcool, from the file libcool-3.14.5.i386.rpm located in the current working directory. Gi as root, to make this happen. Do not use yum or dnf.

root$ c

Correct Answer: **rpm -i libcool-3.14.5.i386.rpm, rpm -iv libcool-3.14.5.i386.rpm, rpm -ivh libcool-3.14.5.i386.rpm**

1. View a text file with less

As showed in the syntax, you can use the less command to view a file in the following fashion:

```
less [option]<filename>
```