File management

- `gzip` and `gunzip`
- `tar`
- `find`
- `df` and `du`
- `od`
- `sftp` and `scp`

`gzip` and `gunzip`

- `gzip` compresses the files named on the command line. After compressing them, it renames them with `.gz` suffixes.
- General form:

  ```
  gzip [FILE]*
  ```
- `gunzip` undoes compression created by `gzip`.
- General form:

  ```
  unzip [FILE]*
  ```
- Other programs that have been used for compression: `compact`, `compress`, and `zip/unzip`.
- You can also use `gzip/gunzip` as filters with the `-c` option, which redirects output to stdout.
- Finally, you can specify the level of compression; `-1` gives the fastest compression but does not optimize space, and `-9` gives the slowest compression but the best use of space.

tar

tar is an old utility, and literally stands for “Tape Archiver”. These days, it is used far more often to handle file archives. It is very useful for creating transportable files between systems, such as when you want to mail a group of files to someone else.

`tar` options

```
-c       # create an archive
-x       # extract from an archive
-t       # shows files in an archive
-f       # specify a file (the default is a tape device!). You can
         # specify stdout with ``-'' (or use -O)
-C       # change directory
-v       # work verbosely
-z       # use gzip/gunzip; if used with -c, creates a gzip'd file; if
         # with -x or -t, it uses gunzip to read the existing archive f
-p       # preserve permissions when extracting
```

Using tar

Typically, you will do something like this to create a tar archive of an existing subdirectory:

```
tar cf DIRNAME.tar [DIRNAME]+
tar czf DIRNAME.tar.gz [DIRNAME]+
tar -c -f DIRNAME.tar [DIRNAME]+
tar -c -z -f  DIRNAME.tgz [DIRNAME]+
```

Using `tar`

Typically, you will do something like this to extract an `tar` archive of an existing subdirectory:

```
tar xf DIRNAME.tar
tar xzf DIRNAME.tar.gz
tar -x -f DIRNAME.tar
tar -x -z -f DIRNAME.tgz
```

find

One of the most useful tools with a recondite syntax is find. It allows you to search a directory for files matching some subset of a large number of possible criteria.

find [PATH]+ CRITERIA

find criteria

```
-name FILENAME          # finds files which match FILENAME, which can co
                        # wildcards
-iname FILENAME         # same as -name, but also case-insensitive
-size [+/-]N[bck]       # very useful, it finds files by size. using 'b'
                        # indicates N is in blocks; using 'c' indicates
                        # using 'k' indicates N using kilobytes. Using '
                        # the file is greater than N in size; using '-'
                        # that it is less than N in size; using neither
                        # the file is exactly size N.
-mtime [+/-]N           # find files based on their last modification ti
                        # in days. +N means match files that have been m
                        # than N days; -N means match files that have be
                        # less than N days; N means match files that hav
                        # exactly N days previous.
-ls                     # show files in {\tt ls} format rather than just
-printf                 # lets you specify arbitrary output formats
-exec COMMAND ;         # lets you run COMMAND over every matching file
-okay COMMAND ;         # same as -exec, but queries you for confirmatio
                        # the command
```

`find` logical operators

```
CRIT1 -a CRIT2        # match only if both criteria CRIT1 and CRIT2 ho
CRIT1 -o CRIT2        # match if either criteria CRIT1 or CRIT2 holds
!CRIT1                # match if criterion CRIT1 does not hold
\( EXPR \)            # evaluate EXPR early
```

find examples

```
find .                  # walk the current directory and its subdirector

find /tmp -mtime +6     # find all files in /tmp that have not been
                        # modified in 6 days

find /tmp -name core -exec rm {} \;     # remove files named 'core' fro

find /tmp -name core -o name '*.o' -okay rm {} \;
                        # query to remove files that are named 'core' or

find /tmp -iname '*.sh' -exec chmod +x {} \;
                        # add execute permission to all files that end i
                        # '.SH', '.Sh', or '.sH'
```

df and du

The df command displays information about mounted filesystems. If you don't specify any, then all of the mounted filesystems are shown. You don't have to specify mount points; any file inside of a filesystem is acceptable:

```
[2006-Fall]$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda2             75766204  19014760  52902672  27% /
/dev/hda1               101089     40221     55649  42% /boot
none                    251668         0    251668   0% /dev/shm
/dev/sda1               981192    480508    450840  52% /mnt-tmp
[2006-Fall]$ df /boot/boot.b
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1               101089     40221     55649  42% /boot
```

df and du

The du command shows you the usage of disk space. With no options, it walks your current directory and shows you the space in blocks used by each subdirectory. With -s, it just shows you a summary. You can force du to display in 1k blocks with the -k option.

```
[2006-Fall]$ du -sk .
8744          .
[2006-Fall]$ du midterm1
80        midterm1/Questions/Shell
20        midterm1/Questions/Process
52        midterm1/Questions/Perl
20        midterm1/Questions/Emacs
20        midterm1/Questions/Awk
20        midterm1/Questions/General
980        midterm1/Questions
1636        midterm1
```

od

The od (octal dump) program writes representations of files to stdout. If '-' is specified, then it looks to stdin for input.
For example, the default od output for the current pdf file is:

```
[langley@sophie 2006-Fall]$ od 22-filemanagement.pdf
0000000 050045 043104 030455 031456 032412 030040 067440 065142
0000020 036012 020074 051457 027440 067507 067524 027440 020104
0000040 033133 030040 051040 020040 043057 072151 056440 037040
0000060 005076 067145 067544 065142 034412 030040 067440 065142
0000100 036040 005074 046057 067145 072147 020150 030465 020064
0000120 020040 020040 020040 027412 064506 072154 071145 027440
 ...
```

od and xxd

od is useful in several ways; for instance, you can find control characters embedded in files that an editor might not display in a reasonable fashion (though emacs is pretty good at displaying embedded characters.)

Using od

```
[2006-Fall]$ od -a 22-filemanagement.pdf
0000000   %   P   D   F   -   1   .   3  nl   5  sp   0  sp   o   b
0000020  nl   <   <  sp   /   S  sp   /   G   o   T   o  sp   /   D   s
0000040   [   6  sp   0  sp   R  sp  sp   /   F   i   t  sp   ]  sp
0000060   >  nl   e   n   d   o   b   j  nl   9  sp   0  sp   o   b
0000100  sp   <   <  nl   /   L   e   n   g   t   h  sp   5   1   4   s
0000120  sp  sp  sp  sp  sp  sp  nl   /   F   i   l   t   e   r  sp
 ....
```

Using od

```
[2006-Fall]$ od -c 22-filemanagement.pdf
0000000   %   P   D   F   -   1   .   3  \n   5       0       o   b
0000020  \n   <   <       /   S       /   G   o   T   o       /   D
0000040   [   6       0       R           /   F   i   t       ]
0000060   >  \n   e   n   d   o   b   j  \n   9       0       o   b
0000100       <   <  \n   /   L   e   n   g   t   h       5   1   4
0000120                          \n   /   F   i   l   t   e   r
 ....
```

Using xxd

The program xxd adds some functionality to od: specifically, it can read a dump and recreate a binary from it. This is very useful for "patching" a binary.

```
[2006-Fall]$ xxd Script12.sh
0000000: 2321 2f62 696e 2f62 6173 680a 0a23 2032  #!/bin/bash..# 2
0000010: 3030 3620 3039 2031 3120 2d20 7264 6c0a  006 09 11 - rdl.
0000020: 666f 7220 6e61 6d65 2069 6e20 2a0a 646f  for name in *.do
0000030: 0a20 2069 6620 5b20 2d66 2022 246e 616d  .  if [ -f "$nam
0000040: 6522 205d 0a20 2074 6865 6e0a 2020 2020  e" ].  then.
0000050: 2065 6368 6f20 2273 6b69 7070 696e 6720   echo "skipping
0000060: 246e 616d 6522 0a20 2020 2020 636f 6e74  $name".     cont
0000070: 696e 7565 0a20 2065 6c73 650a 2020 2020  inue.  else.
0000080: 2065 6368 6f20 2270 726f 6365 7373 2024   echo "process $
0000090: 6e61 6d65 220a 2020 6669 0a64 6f6e 650a  name".  fi.done.

[[ ... edit file to change 0000013 to '37' rather than '36' ... ]]
```

Using xxd

```
[2006-Fall]$ !! > /tmp/xyz
xxd Script12.sh > /tmp/xyz
[2006-Fall]$ xxd -r /tmp/xyz
#!/bin/bash

# 2007 09 11 - rdl
for name in *
do
  if [ -f "$name" ]
  then
     echo "skipping $name"
     continue
  else
     echo "process $name"
  fi
done
```

nm

The `nm` utility lets you print out the namelist of symbols from object files.

This was very useful in finding where a particular variable or function is defined.

[Historical note: Reading the namelist was also a method used "wayback when" to access particular areas of the kernel to make reports on such values as `uptime`. Literally, the program would parse the namelist of the kernel, find the reference to the variable that it wanted, and read that area of memory from `/dev/kernel` to find the values it wanted.]

strip

The strip utility removes optional symbol table, debugging, and line number information from an object file or an executable. strip will reduce the amount of space used by object files and binaries.

sftp

You can transfer files securely with the `sftp` program.

```
sftp [USERNAME]@HOSTNAME
```

Common sftp commands

```
ls [NAME]          # show a directory entry for NAME if specified, other
                   # the present remove working directory
dir [NAME]         # alias for 'ls'
!                  # start a subshell
!ls                # show the local directory (via a subshell)
!COMMAND           # run command in subshell
put LOCALFILE [REMOTEFILE]  # put a local file on the remote machine;
                            # the filename 'REMOTENAME' if specified
get REMOTEFILE [LOCALFILE]  # pull a remote file to the local machine;
                            # it LOCALNAME if specified
cd [DIR]           # change directory on the remote side
lcd [DIR]          # change directory on the local side
chmod PERM FILE    # change permissions on remote file FILE
pwd                # show the current remote directory
lpwd               # show the local directory
mkdir DIR          # create a new directory on the remote side
```

scp

You can also noninteractivley transfer a file or directory with `scp`:

```
[2006-Fall]$ scp /tmp/xyz langley@www.cs.fsu.edu:/tmp/xyz
                       -=-= AUTHORIZED USERS ONLY =-=-
You are attempting to log into a FSU Computer Science Department machi
Please be advised by continuing that you agree to the terms of the
Computer Access and Usage Policy of the Department of Computer Science
                       -=-= AUTHORIZED USERS ONLY =-=-
langley@www.cs.fsu.edu's password: XXXXXXX
[2006-Fall]$ scp -r /etc langley@www.cs.fsu.edu:/tmp/backup-etc
```

sshfs

One of the more exciting developments in the last few years is ability to use FUSE over ssh to mount remote filesystems as an ordinary user. No longer do you have to wait and hope that a system administrator will export a filesystem that you need or want to use. Instead, you can just mount it locally – in fact, that's how this very slide was created, by editing a LaTeXfile remotely!

sshfs, cont’d

The syntax for this is:

```
sshfs REMOTEUSERNAME@REMOTEMACHINE:/REMOTE/DIR/PATH /LOCAL/PATH
```