Basic Shell Usage, part 2

ComS 252 — Iowa State University

Andrew Miner

Viewing file contents

`cat`: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$
```

Viewing file contents

`cat`: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
```

Viewing file contents

cat: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
prompt$
```

Viewing file contents

cat: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
prompt$ cat /bin/ls
```

Viewing file contents

cat: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
prompt$ cat /bin/ls
          AÃÿÿ:c
```

Viewing file contents

cat: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
prompt$ cat /bin/ls
          AÃÿÿ:cX•:  ??Öp$8(
                      4?G□<π
```

Viewing file contents

cat: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
prompt$ cat /bin/ls
          AÃÿÿ:cX•:  ??Öp$8(
                       4?G□<πw
E)L3X$ËÑz   proμpt$
```

Viewing file contents

cat: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
prompt$ cat /bin/ls
          AÃÿÿ:cX•:  ??Öp$8(
                       4?G□<πw
E)L3X$ËÑz   proμpt$ μμμ
```

Viewing file contents

cat: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
prompt$ cat /bin/ls
          AÃÿÿ:cX•:  ??Öp$8(
                       4?G□<πw
E)L3X$ËÑz   proμpt$ μμμ
-baÿh:  μμμ:  coμμa□d □ot fou□d
proμpt$
```

Viewing file contents

cat: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain
prompt$ cat /bin/ls
          AÃÿÿ:cX•:  ??Öp$8(
                       4?G□<πw
E)L3X$ËÑz   proμpt$ μμμ
-baÿh:  μμμ:  coμμa□d □ot fou□d
proμpt$ reÿet
```

Viewing file contents

`cat`: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

Viewing file contents

`cat`: concatenate files

- Displays contents of files (arguments) as ASCII text
- Binary files will display as "garbage"

```
prompt$
```

Viewing file contents (2)

`hexdump`: hexadecimal dump a file

- Displays a file in various formats

`-C` : "canonical" display, hexadecimal and ASCII

- Each line: 16 characters of the file, in columns
  1. Hexadecimal byte count
  2. Hexadecimal encoding of the 16 characters
  3. ASCII encoding of the 16 characters (if printable)

```
prompt$
```

Viewing file contents (2)

`hexdump`: hexadecimal dump a file

- Displays a file in various formats

`-C` : "canonical" display, hexadecimal and ASCII

- Each line: 16 characters of the file, in columns
  1. Hexadecimal byte count
  2. Hexadecimal encoding of the 16 characters
  3. ASCII encoding of the 16 characters (if printable)

```
prompt$ hexdump -C /etc/hosts
```

Viewing file contents (2)

`hexdump`: hexadecimal dump a file

- Displays a file in various formats

`-C` : "canonical" display, hexadecimal and ASCII

- Each line: 16 characters of the file, in columns
  1. Hexadecimal byte count
  2. Hexadecimal encoding of the 16 characters
  3. ASCII encoding of the 16 characters (if printable)

```
prompt$ hexdump -C /etc/hosts
00000000  31 32 37 2e 30 2e 30 2e  31 20 20 20 6c 6f 63 61  |127.0.0.1   loca|
00000010  6c 68 6f 73 74 20 6c 6f  63 61 6c 68 6f 73 74 2e  |lhost localhost.|
00000020  6c 6f 63 61 6c 64 6f 6d  61 69 6e 0a 3a 3a 31 20  |localdomain.::1 |
00000030  20 20 20 20 20 20 20 20  6c 6f 63 61 6c 68 6f 73  |        localhos|
00000040  74 20 6c 6f 63 61 6c 68  6f 73 74 2e 6c 6f 63 61  |t localhost.loca|
00000050  6c 64 6f 6d 61 69 6e 0a                           |ldomain.|
00000058
prompt$
```

Viewing file contents (2)

`hexdump`: hexadecimal dump a file

- Displays a file in various formats

`-C` : "canonical" display, hexadecimal and ASCII

- Each line: 16 characters of the file, in columns
  1. Hexadecimal byte count
  2. Hexadecimal encoding of the 16 characters
  3. ASCII encoding of the 16 characters (if printable)

```
prompt$ hexdump -C /etc/hosts
00000000  31 32 37 2e 30 2e 30 2e  31 20 20 20 6c 6f 63 61  |127.0.0.1   loca|
00000010  6c 68 6f 73 74 20 6c 6f  63 61 6c 68 6f 73 74 2e  |lhost localhost.|
00000020  6c 6f 63 61 6c 64 6f 6d  61 69 6e 0a 3a 3a 31 20  |localdomain.::1 |
00000030  20 20 20 20 20 20 20 20  6c 6f 63 61 6c 68 6f 73  |        localhos|
00000040  74 20 6c 6f 63 61 6c 68  6f 73 74 2e 6c 6f 63 61  |t localhost.loca|
00000050  6c 64 6f 6d 61 69 6e 0a                           |ldomain.|
00000058
prompt$ hexdump -C /bin/ls
```

Viewing file contents (2)

`hexdump`: hexadecimal dump a file

- Displays a file in various formats

`-C` : "canonical" display, hexadecimal and ASCII

- Each line: 16 characters of the file, in columns
  1. Hexadecimal byte count
  2. Hexadecimal encoding of the 16 characters
  3. ASCII encoding of the 16 characters (if printable)

```
0001c820  00 53 06 08 e8 c2 01 00  00 0c 00 00 00 00 00 00  |.S..............|
0001c830  00 00 00 00 20 00 00 00  00 00 00 00 5d 00 00 00  |.... .......]...|
0001c840  01 00 00 00 00 00 00 00  00 00 00 00 e8 c2 01 00  |................|
0001c850  10 00 00 00 00 00 00 00  00 00 00 00 04 00 00 00  |................|
0001c860  00 00 00 00 0f 00 00 00  03 00 00 00 00 00 00 00  |................|
0001c870  00 00 00 00 f8 c2 01 00  09 01 00 00 00 00 00 00  |................|
0001c880  00 00 00 00 01 00 00 00  00 00 00 00              |............|
0001c88c
prompt$
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$ cat foo.txt
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$ cat foo.txt
This is a simple text file
prompt$
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$ cat foo.txt
This is a simple text file
prompt$ cp foo.txt bar.txt
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$ cat foo.txt
This is a simple text file
prompt$ cp foo.txt bar.txt
prompt$
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$ cat foo.txt
This is a simple text file
prompt$ cp foo.txt bar.txt
prompt$ ls
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$ cat foo.txt
This is a simple text file
prompt$ cp foo.txt bar.txt
prompt$ ls
bar.txt    foo.txt
prompt$
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$ cat foo.txt
This is a simple text file
prompt$ cp foo.txt bar.txt
prompt$ ls
bar.txt    foo.txt
prompt$ cat bar.txt
```

Copying (1)

cp: copy files

Usage:

1. `cp source target`

   source : an existing file

   target : name for the copy

```
prompt$ cat foo.txt
This is a simple text file
prompt$ cp foo.txt bar.txt
prompt$ ls
bar.txt    foo.txt
prompt$ cat bar.txt
This is a simple text file
prompt$
```

Copying (2)

cp: copy files

Usage:

2. `cp src1 src2 ...srcn Directory`

   src1 : an existing file

   ⋮

   src*n* : an existing file

   Directory : An existing directory

   - Copies each source file into `Directory`
   - The copy has the same name as the original

Example: cp into a directory

Example: cp into a directory

```
prompt$ ls
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$ mkdir Copies
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$ mkdir Copies
prompt$
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$ mkdir Copies
prompt$ cp bar.txt crud.txt foo.txt Copies
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$ mkdir Copies
prompt$ cp bar.txt crud.txt foo.txt Copies
prompt$
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$ mkdir Copies
prompt$ cp bar.txt crud.txt foo.txt Copies
prompt$ ls
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$ mkdir Copies
prompt$ cp bar.txt crud.txt foo.txt Copies
prompt$ ls
bar.txt     Copies      crud.txt    foo.txt
prompt$
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$ mkdir Copies
prompt$ cp bar.txt crud.txt foo.txt Copies
prompt$ ls
bar.txt     Copies      crud.txt    foo.txt
prompt$ ls Copies
```

Example: cp into a directory

```
prompt$ ls
bar.txt     crud.txt    foo.txt
prompt$ mkdir Copies
prompt$ cp bar.txt crud.txt foo.txt Copies
prompt$ ls
bar.txt     Copies      crud.txt    foo.txt
prompt$ ls Copies
bar.txt     crud.txt    foo.txt
prompt$
```

Useful switches for cp

-i : interactive
- Ask before overwriting any existing target file

-p : preserve
- Preserves the modification time (and other attributes)
- Otherwise, the copy has the current time

-R : recursive
- If any source is a directory, recursively copies it
- Otherwise, source directories are skipped

What about: `cp -R dir1 dir2`

What about: `cp -R dir1 dir2`

1. If `dir2` does not exist:
   - Creates `dir2`
   - Recursively copies files from `dir1` into `dir2`

What about: `cp -R dir1 dir2`

1. If `dir2` does not exist:
   - Creates `dir2`
   - Recursively copies files from `dir1` into `dir2`
2. If `dir2` does exist:
   - Makes copy of `dir1` inside `dir2`

Moving (1)

`mv`: move files

Usage (just like `cp`):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$
```

Moving (1)

`mv`: move files

Usage (just like `cp`):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$ cat foo.txt
```

Moving (1)

mv: move files

Usage (just like cp):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$ cat foo.txt
This is a simple text file
prompt$
```

Moving (1)

`mv`: move files

Usage (just like `cp`):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$ cat foo.txt
This is a simple text file
prompt$ mv foo.txt bar.txt
```

Moving (1)

mv: move files

Usage (just like cp):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$ cat foo.txt
This is a simple text file
prompt$ mv foo.txt bar.txt
prompt$
```

Moving (1)

`mv`: move files

Usage (just like `cp`):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$ cat foo.txt
This is a simple text file
prompt$ mv foo.txt bar.txt
prompt$ ls
```

Moving (1)

mv: move files

Usage (just like cp):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$ cat foo.txt
This is a simple text file
prompt$ mv foo.txt bar.txt
prompt$ ls
bar.txt
prompt$
```

Moving (1)

mv: move files

Usage (just like cp):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$ cat foo.txt
This is a simple text file
prompt$ mv foo.txt bar.txt
prompt$ ls
bar.txt
prompt$ cat bar.txt
```

Moving (1)

mv: move files

Usage (just like cp):

1. `mv source target`

   source : an existing file

   target : new name for the file

```
prompt$ cat foo.txt
This is a simple text file
prompt$ mv foo.txt bar.txt
prompt$ ls
bar.txt
prompt$ cat bar.txt
This is a simple text file
prompt$
```

Moving (2)

`mv`: move files

Usage (just like `cp`):

2. `mv src1 src2 ...srcn Directory`

   src1 : an existing file or directory

   ⋮

   src*n* : an existing file or directory

   Directory : An existing directory

   - Moves each source item into `Directory`

```
prompt$
```

Moving (2)

`mv`: move files

Usage (just like `cp`):

2. `mv src1 src2 ...srcn Directory`

   src1 : an existing file or directory

   ⋮

   src*n* : an existing file or directory

   Directory : An existing directory

   - Moves each source item into `Directory`

```
prompt$ mv rhel01.iso rhel02.iso rhel03.iso /tmp
```

Moving (2)

`mv`: move files

Usage (just like `cp`):

2. `mv src1 src2 ...srcn Directory`

   src1 : an existing file or directory

   ⋮

   src*n* : an existing file or directory

   Directory : An existing directory

   - Moves each source item into `Directory`

```
prompt$ mv rhel01.iso rhel02.iso rhel03.iso /tmp
prompt$
```

Moving (2)

`mv`: move files

Usage (just like `cp`):

2. `mv src1 src2 ...srcn Directory`

   src1 : an existing file or directory

   ⋮

   src*n* : an existing file or directory

   Directory : An existing directory

   - Moves each source item into `Directory`

```
prompt$ mv rhel01.iso rhel02.iso rhel03.iso /tmp
prompt$ ls /tmp/
```

Moving (2)

`mv`: move files

Usage (just like `cp`):

2. `mv src1 src2 ...srcn Directory`

   src1 : an existing file or directory

   ⋮

   src*n* : an existing file or directory

   Directory : An existing directory

   - Moves each source item into `Directory`

```
prompt$ mv rhel01.iso rhel02.iso rhel03.iso /tmp
prompt$ ls /tmp/
oldthing.txt  rhel01.iso    rhel02.iso    rhel03.iso
prompt$
```

Removing

`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

Example: `rm`

```
prompt$
```

Example: `rm`

```
prompt$ ls -aF
```

Example: `rm`

```
prompt$ ls -aF
./          .bashrc   foo.cc    .history  .viminfo
../         core      hello.c   math168/
a.out*      cs229/    hello.h   se101/
bar.cc      cs252/    hello.o   .ssh/
prompt$
```

Example: `rm`

```
prompt$ ls -aF
./          .bashrc   foo.cc     .history  .viminfo
../         core      hello.c    math168/
a.out*      cs229/    hello.h    se101/
bar.cc      cs252/    hello.o    .ssh/
prompt$ rm -i a.out core hello.o
```

Example: `rm`

```
prompt$ ls -aF
./          .bashrc   foo.cc    .history  .viminfo
../         core      hello.c   math168/
a.out*      cs229/    hello.h   se101/
bar.cc      cs252/    hello.o   .ssh/
prompt$ rm -i a.out core hello.o
remove a.out?
```

Example: `rm`

```
prompt$ ls -aF
./          .bashrc   foo.cc    .history  .viminfo
../         core      hello.c   math168/
a.out*      cs229/    hello.h   se101/
bar.cc      cs252/    hello.o   .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
```

Example: `rm`

```
prompt$ ls -aF
./          .bashrc   foo.cc    .history  .viminfo
../         core      hello.c   math168/
a.out*      cs229/    hello.h   se101/
bar.cc      cs252/    hello.o   .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core?
```

Example: `rm`

```
prompt$ ls -aF
./          .bashrc   foo.cc     .history  .viminfo
../         core      hello.c    math168/
a.out*      cs229/    hello.h    se101/
bar.cc      cs252/    hello.o    .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core? y
```

Example: rm

```
prompt$ ls -aF
./          .bashrc   foo.cc    .history  .viminfo
../         core      hello.c   math168/
a.out*      cs229/    hello.h   se101/
bar.cc      cs252/    hello.o   .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core? y
remove hello.o?
```

Example: rm

```
prompt$ ls -aF
./          .bashrc   foo.cc    .history  .viminfo
../         core      hello.c   math168/
a.out*      cs229/    hello.h   se101/
bar.cc      cs252/    hello.o   .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core? y
remove hello.o? y
```

Example: `rm`

```
prompt$ ls -aF
./          .bashrc   foo.cc    .history  .viminfo
../         core      hello.c   math168/
a.out*      cs229/    hello.h   se101/
bar.cc      cs252/    hello.o   .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core? y
remove hello.o? y
prompt$
```

Example: rm

```
prompt$ ls -aF
./          .bashrc   foo.cc     .history  .viminfo
../         core      hello.c    math168/
a.out*      cs229/    hello.h    se101/
bar.cc      cs252/    hello.o    .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core? y
remove hello.o? y
prompt$ rm -r foo.cc bar.cc se101
```

Example: rm

```
prompt$ ls -aF
./          .bashrc   foo.cc     .history  .viminfo
../         core      hello.c    math168/
a.out*      cs229/    hello.h    se101/
bar.cc      cs252/    hello.o    .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core? y
remove hello.o? y
prompt$ rm -r foo.cc bar.cc se101
prompt$
```

Example: rm

```
prompt$ ls -aF
./          .bashrc   foo.cc     .history  .viminfo
../         core      hello.c    math168/
a.out*      cs229/    hello.h    se101/
bar.cc      cs252/    hello.o    .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core? y
remove hello.o? y
prompt$ rm -r foo.cc bar.cc se101
prompt$ ls -aF
```

Example: `rm`

```
prompt$ ls -aF
./          .bashrc   foo.cc    .history  .viminfo
../         core      hello.c   math168/
a.out*      cs229/    hello.h   se101/
bar.cc      cs252/    hello.o   .ssh/
prompt$ rm -i a.out core hello.o
remove a.out? n
remove core? y
remove hello.o? y
prompt$ rm -r foo.cc bar.cc se101
prompt$ ls -aF
./          .bashrc   hello.c   math168/
../         cs229/    hello.h   .ssh/
a.out*      cs252/    .history  .viminfo
prompt$
```

Un-removing a file

If I `rm` a file by mistake, can I get it back?

Un-removing a file

If I `rm` a file by mistake, can I get it back?

Short answer: NO

So be careful with `rm`

Un-removing a file

If I `rm` a file by mistake, can I get it back?

Short answer: NO

So be careful with `rm`

Long answer: maybe

- Depends on the filesystem and how files are deleted
- There are undelete utilities (by filesystem type)
- Need to (cleanly) disconnect the disk right away!
  - Prevents the deleted file from being overwritten

Un-removing a file

If I `rm` a file by mistake, can I get it back?

Short answer: NO

So be careful with `rm`

Long answer: maybe

- Depends on the filesystem and how files are deleted
- There are undelete utilities (by filesystem type)
- Need to (cleanly) disconnect the disk right away!
  - Prevents the deleted file from being overwritten

System administrator answer

I can get you a copy from last night's backup

Using *

We can list out several files using *

- Will generate a list of matching files
- Means, “fill in with zero or more characters”
- Usually does not match a leading .
- Can be used with any utility that takes lists of files

Examples with *

- Alice's pictures (start with `IMG`) to `~alice/images`
- Bob's pictures (start with `DSC`) to `~bob/images`
- Remove all other files

```
prompt$
```

Examples with *

- Alice's pictures (start with `IMG`) to `~alice/images`
- Bob's pictures (start with `DSC`) to `~bob/images`
- Remove all other files

```
prompt$ ls -aF --color
```

Examples with *

- Alice's pictures (start with `IMG`) to `~alice/images`
- Bob's pictures (start with `DSC`) to `~bob/images`
- Remove all other files

```
prompt$ ls -aF --color
./              DSC_547.jpg   DSC_552.json  IMG_766.jpeg
../             DSC_547.json  DSC_553.jpg   IMG_767.jpeg
.catalog        DSC_549.jpg   DSC_553.json  IMG_768.jpeg
DSC_544.jpg     DSC_549.json  IMG_760.jpeg  IMG_769.jpeg
DSC_544.json    DSC_550.jpg   IMG_761.jpeg  IMG_770.jpeg
DSC_545.jpg     DSC_550.json  IMG_762.jpeg  IMG_771.jpeg
DSC_545.json    DSC_551.jpg   IMG_763.jpeg  VID_548.mov
DSC_546.jpg     DSC_551.json  IMG_764.jpeg
DSC_546.json    DSC_552.jpg   IMG_765.jpeg
prompt$
```

Examples with *

- Alice's pictures (start with IMG) to ~alice/images
- Bob's pictures (start with DSC) to ~bob/images
- Remove all other files

```
prompt$ ls -aF --color
./              DSC_547.jpg   DSC_552.json  IMG_766.jpeg
../             DSC_547.json  DSC_553.jpg   IMG_767.jpeg
.catalog        DSC_549.jpg   DSC_553.json  IMG_768.jpeg
DSC_544.jpg     DSC_549.json  IMG_760.jpeg  IMG_769.jpeg
DSC_544.json    DSC_550.jpg   IMG_761.jpeg  IMG_770.jpeg
DSC_545.jpg     DSC_550.json  IMG_762.jpeg  IMG_771.jpeg
DSC_545.json    DSC_551.jpg   IMG_763.jpeg  VID_548.mov
DSC_546.jpg     DSC_551.json  IMG_764.jpeg
DSC_546.json    DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
```

Examples with *

- Alice's pictures (start with `IMG`) to `~alice/images`
- Bob's pictures (start with `DSC`) to `~bob/images`
- Remove all other files

```
./              DSC_547.jpg   DSC_552.json  IMG_766.jpeg
../             DSC_547.json  DSC_553.jpg   IMG_767.jpeg
.catalog        DSC_549.jpg   DSC_553.json  IMG_768.jpeg
DSC_544.jpg     DSC_549.json  IMG_760.jpeg  IMG_769.jpeg
DSC_544.json    DSC_550.jpg   IMG_761.jpeg  IMG_770.jpeg
DSC_545.jpg     DSC_550.json  IMG_762.jpeg  IMG_771.jpeg
DSC_545.json    DSC_551.jpg   IMG_763.jpeg  VID_548.mov
DSC_546.jpg     DSC_551.json  IMG_764.jpeg
DSC_546.json    DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
prompt$
```

Examples with *

- Alice's pictures (start with `IMG`) to `~alice/images`
- Bob's pictures (start with `DSC`) to `~bob/images`
- Remove all other files

```
./              DSC_547.jpg   DSC_552.json  IMG_766.jpeg
../             DSC_547.json  DSC_553.jpg   IMG_767.jpeg
.catalog        DSC_549.jpg   DSC_553.json  IMG_768.jpeg
DSC_544.jpg     DSC_549.json  IMG_760.jpeg  IMG_769.jpeg
DSC_544.json    DSC_550.jpg   IMG_761.jpeg  IMG_770.jpeg
DSC_545.jpg     DSC_550.json  IMG_762.jpeg  IMG_771.jpeg
DSC_545.json    DSC_551.jpg   IMG_763.jpeg  VID_548.mov
DSC_546.jpg     DSC_551.json  IMG_764.jpeg
DSC_546.json    DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
prompt$ mv DSC*.jpg ~bob/images
```

Examples with *

- Alice's pictures (start with IMG) to ~alice/images
- Bob's pictures (start with DSC) to ~bob/images
- Remove all other files

```
../             DSC_547.json  DSC_553.jpg   IMG_767.jpeg
.catalog        DSC_549.jpg   DSC_553.json  IMG_768.jpeg
DSC_544.jpg     DSC_549.json  IMG_760.jpeg  IMG_769.jpeg
DSC_544.json    DSC_550.jpg   IMG_761.jpeg  IMG_770.jpeg
DSC_545.jpg     DSC_550.json  IMG_762.jpeg  IMG_771.jpeg
DSC_545.json    DSC_551.jpg   IMG_763.jpeg  VID_548.mov
DSC_546.jpg     DSC_551.json  IMG_764.jpeg
DSC_546.json    DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
prompt$ mv DSC*.jpg ~bob/images
prompt$
```

Examples with *

- Alice's pictures (start with IMG) to ~alice/images
- Bob's pictures (start with DSC) to ~bob/images
- Remove all other files

```
../             DSC_547.json  DSC_553.jpg   IMG_767.jpeg
.catalog        DSC_549.jpg   DSC_553.json  IMG_768.jpeg
DSC_544.jpg     DSC_549.json  IMG_760.jpeg  IMG_769.jpeg
DSC_544.json    DSC_550.jpg   IMG_761.jpeg  IMG_770.jpeg
DSC_545.jpg     DSC_550.json  IMG_762.jpeg  IMG_771.jpeg
DSC_545.json    DSC_551.jpg   IMG_763.jpeg  VID_548.mov
DSC_546.jpg     DSC_551.json  IMG_764.jpeg
DSC_546.json    DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
prompt$ mv DSC*.jpg ~bob/images
prompt$ ls -af --color
```

Examples with *

- Alice's pictures (start with `IMG`) to `~alice/images`
- Bob's pictures (start with `DSC`) to `~bob/images`
- Remove all other files

```
DSC_545.json  DSC_551.jpg   IMG_763.jpeg  VID_548.mov
DSC_546.jpg   DSC_551.json  IMG_764.jpeg
DSC_546.json  DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
prompt$ mv DSC*.jpg ~bob/images
prompt$ ls -af --color
./            DSC_545.json  DSC_550.json  VID_548.mov
../           DSC_546.json  DSC_551.json
.catalog      DSC_547.json  DSC_552.json
DSC_544.json  DSC_549.json  DSC_553.json
prompt$
```

Examples with *

- Alice's pictures (start with IMG) to ~alice/images
- Bob's pictures (start with DSC) to ~bob/images
- Remove all other files

```
DSC_545.json  DSC_551.jpg   IMG_763.jpeg  VID_548.mov
DSC_546.jpg   DSC_551.json  IMG_764.jpeg
DSC_546.json  DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
prompt$ mv DSC*.jpg ~bob/images
prompt$ ls -af --color
./            DSC_545.json  DSC_550.json  VID_548.mov
../           DSC_546.json  DSC_551.json
.catalog      DSC_547.json  DSC_552.json
DSC_544.json  DSC_549.json  DSC_553.json
prompt$ rm *
```

Examples with *

- Alice's pictures (start with IMG) to ~alice/images
- Bob's pictures (start with DSC) to ~bob/images
- Remove all other files

```
DSC_546.jpg   DSC_551.json  IMG_764.jpeg
DSC_546.json  DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
prompt$ mv DSC*.jpg ~bob/images
prompt$ ls -af --color
./            DSC_545.json  DSC_550.json  VID_548.mov
../           DSC_546.json  DSC_551.json
.catalog      DSC_547.json  DSC_552.json
DSC_544.json  DSC_549.json  DSC_553.json
prompt$ rm *
prompt$
```

Examples with *

- Alice's pictures (start with `IMG`) to `~alice/images`
- Bob's pictures (start with `DSC`) to `~bob/images`
- Remove all other files

```
DSC_546.jpg   DSC_551.json  IMG_764.jpeg
DSC_546.json  DSC_552.jpg   IMG_765.jpeg
prompt$ mv IMG* ~alice/images
prompt$ mv DSC*.jpg ~bob/images
prompt$ ls -af --color
./            DSC_545.json  DSC_550.json  VID_548.mov
../           DSC_546.json  DSC_551.json
.catalog      DSC_547.json  DSC_552.json
DSC_544.json  DSC_549.json  DSC_553.json
prompt$ rm *
prompt$ ls -af --color
```

Examples with *

- Alice's pictures (start with `IMG`) to `~alice/images`
- Bob's pictures (start with `DSC`) to `~bob/images`
- Remove all other files

```
prompt$ mv IMG* ~alice/images
prompt$ mv DSC*.jpg ~bob/images
prompt$ ls -af --color
./              DSC_545.json  DSC_550.json  VID_548.mov
../             DSC_546.json  DSC_551.json
.catalog        DSC_547.json  DSC_552.json
DSC_544.json    DSC_549.json  DSC_553.json
prompt$ rm *
prompt$ ls -af --color
./   ../   .catalog
prompt$
```

Who can do what

- Until now, we have ignored a very important question:

Who can do what

- Until now, we have ignored a very important question:
- Who is allowed to do that?

Who can do what

- Until now, we have ignored a very important question:
- Who is allowed to do that?
- E.g., Is `chuck` allowed to cat `alice`'s files?

Who can do what

- Until now, we have ignored a very important question:
- Who is allowed to do that?
- E.g., Is `chuck` allowed to cat `alice`'s files?
- E.g., Is `bob` allowed to remove `alice`'s files?

Who can do what

- Until now, we have ignored a very important question:
- Who is allowed to do that?
- E.g., Is `chuck` allowed to cat `alice`'s files?
- E.g., Is `bob` allowed to remove `alice`'s files?
- E.g., Can `bob` create a file in `alice`'s home directory?

Who can do what

- Until now, we have ignored a very important question:
- Who is allowed to do that?
- E.g., Is `chuck` allowed to cat `alice`'s files?
- E.g., Is `bob` allowed to remove `alice`'s files?
- E.g., Can `bob` create a file in `alice`'s home directory?
- UNIX handles these questions using file permissions

File permissions

- First column of "`ls -l`" gives the file permissions
- Remember: first character gives the file type
- Next 3 characters: can the file owner

slot 2:	Read the file?	`r`: yes	`-`: no
slot 3:	Modify the file?	`w`: yes	`-`: no
slot 4:	Execute the file?	`x`: yes	`-`: no

- Next 3 characters: can members of the file group

slot 5:	Read the file?	`r`: yes	`-`: no
slot 6:	Modify the file?	`w`: yes	`-`: no
slot 7:	Execute the file?	`x`: yes	`-`: no

- Last 3 characters: can everyone else

slot 8:	Read the file?	`r`: yes	`-`: no
slot 9:	Modify the file?	`w`: yes	`-`: no
slot 10:	Execute the file?	`x`: yes	`-`: no

Permission example

```
prompt$ ls -l foo.txt
-rw-r----- 1  alice  hackers  2995  Feb 04 1999  foo.txt
prompt$
```

For the file `foo.txt`:

- The file is owned by `alice`
- `alice` may read and write the file, but not execute it
- The file group is `hackers`
- If `bob` is a member of group `hackers`,
  then `bob` may read the file, but not write or execute it
- If `bob` is not a member of group `hackers`,
  then `bob` may not read, write, or execute the file

Permission example

```
prompt$ ls -l foo.txt
-rw-r----- 1  alice  hackers  2995  Feb 04 1999  foo.txt
prompt$
```

For the file `foo.txt`:

- The file is owned by `alice`
- `alice` may read and write the file, but not execute it
- The file group is `hackers`
- If `bob` is a member of group `hackers`,
  then `bob` may read the file, but not write or execute it
- If `bob` is not a member of group `hackers`,
  then `bob` may not read, write, or execute the file

Permission example

```
prompt$ ls -l foo.txt
-rw-r----- 1  alice  hackers  2995  Feb 04 1999  foo.txt
prompt$
```

For the file `foo.txt`:

- The file is owned by `alice`
- `alice` may read and write the file, but not execute it
- The file group is `hackers`
- If `bob` is a member of group `hackers`,
  then `bob` may read the file, but not write or execute it
- If `bob` is not a member of group `hackers`,
  then `bob` may not read, write, or execute the file

Permission example

```
prompt$ ls -l foo.txt
-rw-r----- 1  alice  hackers  2995  Feb 04 1999  foo.txt
prompt$
```

For the file `foo.txt`:

- The file is owned by `alice`
- `alice` may read and write the file, but not execute it
- The file group is `hackers`
- If `bob` is a member of group `hackers`,
  then `bob` may read the file, but not write or execute it
- If `bob` is not a member of group `hackers`,
  then `bob` may not read, write, or execute the file

Permission example

```
prompt$ ls -l foo.txt
-rw-r----- 1  alice  hackers  2995  Feb 04 1999  foo.txt
prompt$
```

For the file `foo.txt`:

- The file is owned by `alice`
- `alice` may read and write the file, but not execute it
- The file group is `hackers`
- If `bob` is a member of group `hackers`,
  then `bob` may read the file, but not write or execute it
- If `bob` is not a member of group `hackers`,
  then `bob` may not read, write, or execute the file

Permission example

```
prompt$ ls -l foo.txt
-rw-r------ 1  alice  hackers  2995  Feb 04 1999  foo.txt
prompt$
```

For the file `foo.txt`:

- The file is owned by `alice`
- `alice` may read and write the file, but not execute it
- The file group is `hackers`
- If `bob` is a member of group `hackers`,
  then `bob` may read the file, but not write or execute it
- If `bob` is not a member of group `hackers`,
  then `bob` may not read, write, or execute the file

Permission meaning

For files

read : necessary to view or copy a file

write : necessary to modify a file

execute : necessary to execute a file

For directories

read : necessary to examine entries ("`ls`" the directory)

write : necessary to modify the directory

- Create a file
- Rename a file
- Remove a file

execute : necessary to access a directory ("cd" it)

Necessary permissions for some examples

```
cp src dest
```

```
mv src dest
```

```
rm src
```

Necessary permissions for some examples

`cp src dest`

- Need read permission for `src`

`mv src dest`

`rm src`

Necessary permissions for some examples

`cp src dest`

- Need read permission for `src`
- Need write permission for working directory
  - Otherwise we cannot create file `dest`

`mv src dest`

`rm src`

Necessary permissions for some examples

`cp src dest`

- Need read permission for `src`
- Need write permission for working directory
  - Otherwise we cannot create file `dest`

`mv src dest`

- Need write permission for working directory

`rm src`

Necessary permissions for some examples

`cp src dest`

- Need read permission for `src`
- Need write permission for working directory
  - Otherwise we cannot create file `dest`

`mv src dest`

- Need write permission for working directory

`rm src`

- Need write permission for working directory

Superuser

User account `root` is the "superuser" account

- File permissions do not apply
- `root` can read, modify, execute, rename, delete any file
- `root` can pretty much do anything on the system
- This is necessary for system administration
  - Backing up files
  - Installing new disks
  - etc. . .

Changing permissions

`chmod`: change file permissions

Usage:

1. `chmod [ugoa][-+=][rwx] file1 file2 ... filen`

   - `u` : just the user (file owner) permissions
   - `g` : just the group permissions
   - `o` : just the other (everyone else) permissions
     - o does not mean "owner"
   - `a` : all (user, group, and other)
   - `-` : turn off permissions
   - `+` : turn on permissions
   - `=` : set exactly permissions

Examples: chmod 1

```
chmod ug+r foo.txt
```

Examples: chmod 1

```
chmod ug+r foo.txt
```

Turns on read permission for user and group

```
chmod u=rw foo.txt
```

Examples: chmod 1

```
chmod ug+r foo.txt
```

Turns on read permission for user and group

```
chmod u=rw foo.txt
```

Sets user permissions to `rw-`

```
chmod a-x foo.txt
```

Examples: chmod 1

```
chmod ug+r foo.txt
```

Turns on read permission for user and group

```
chmod u=rw foo.txt
```

Sets user permissions to `rw-`

```
chmod a-x foo.txt
```

Turn off execute permission for user, group, and other

Changing permissions (2)

`chmod`: change file permissions

Usage:

2. `chmod [mode] file1 file2 ... filen`

   mode : three octal digits

   - First digit: user permissions
   - Second digit: group permissions
   - Third digit: other permissions
   - For each digit

     read : add 4
     write : add 2
     execute : add 1

Examples: chmod 2

```
chmod 640 foo.txt
```

Examples: chmod 2

```
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/
```

Examples: chmod 2

```
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

What about publicly writeable directories?

- Consider `/tmp`: preferred place for temporary files
- Files are automatically removed
- Want anyone able to put files there
  - Everyone should have write access to `/tmp`
  - But that means anyone can delete any file

What about publicly writeable directories?

- Consider `/tmp`: preferred place for temporary files
- Files are automatically removed
- Want anyone able to put files there
  - Everyone should have write access to `/tmp`
  - But that means anyone can delete any file
- Solution: set the "sticky bit"
  - Only the file owner (or root) may delete a file
- To set this up for `/tmp`:

```
prompt$ chmod 777 /tmp
prompt$ chmod +t /tmp
prompt$ ls -alF /tmp
total 98
drwxrwxrwt  6 root  root  4096 Jan  1 2000  ./
:
prompt$
```

Who may `chmod` a file?

1. `root`

Who may `chmod` a file?

1. `root`
2. The file owner

Who may `chmod` a file?

1. `root`
2. The file owner
3. Nobody else

Changing the group or owner of a file

`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

Who can `chown` or `chgrp` a file?

Who can change the owner of a file?

1. `root`

Who can `chown` or `chgrp` a file?

Who can change the owner of a file?

1. `root`
2. Nobody else

Who can change the group of a file?

1. `root`

Who can `chown` or `chgrp` a file?

Who can change the owner of a file?

1. `root`
2. Nobody else

Who can change the group of a file?

1. `root`
2. The file owner

Who can `chown` or `chgrp` a file?

Who can change the owner of a file?

1. `root`
2. Nobody else

Who can change the group of a file?

1. `root`
2. The file owner
3. Nobody else

What group can it be changed to?

Who can `chown` or `chgrp` a file?

Who can change the owner of a file?

1. `root`
2. Nobody else

Who can change the group of a file?

1. `root`
2. The file owner
3. Nobody else

What group can it be changed to?

- `root`: anything

Who can `chown` or `chgrp` a file?

Who can change the owner of a file?

1. `root`
2. Nobody else

Who can change the group of a file?

1. `root`
2. The file owner
3. Nobody else

What group can it be changed to?

- `root`: anything
- Ordinary user: any group the user belongs to

Summary of today's commands

`cat` : Concatenate a file (to the display).

`chgrp` : Change file group.

`chmod` : Change permissions.

`chown` : Change file owner.

`cp` : Copy files or directories.

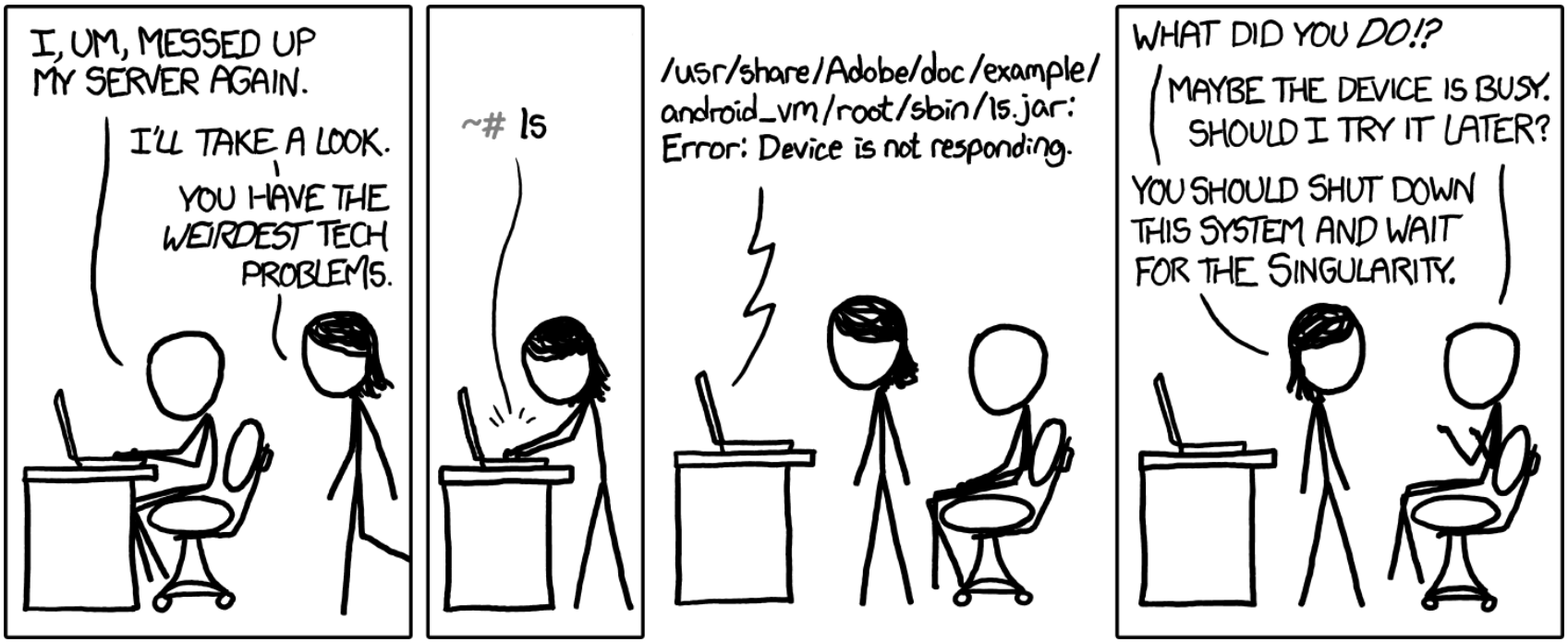
`hexdump` : Show hex contents of a file.

`mv` : Move files or directories.

`reset` : Reset a trashed terminal.

`rm` : Remove files or directories.

An appropriate xkcd comic: `https://xkcd.com/1084`



End of lecture