Long listing columns

1. 10 "mystery characters"
   - First character specifies the file type
     - `-` : Ordinary file
     - `b/c` : block/character device (to be discussed)
     - `d` : directory
     - `l` : symlink (to be discussed)
     - `s` : socket
   - In UNIX, everything is a file
   - Other 9 characters are for permissions (to be discussed)
2. an integer; we will get to that
3. user who owns the file
4. group of the file
   - Allows groups of people to work together
   - Details when we discuss "permissions"

Long listing columns, ctd.

5. Size of the file, in bytes.
   - For devices: the major and minor number
     - Don't worry about what that means
6,7,8. Date and time of last modification.
   - If the file is old, the time is replaced by the year
9. Name of the file.

Example long listing

```
prompt$ ls -lF
total 96
-rwx------ 1 alice hackers  6425 May  4 09:47 a.out*
-rw-r----- 1 alice hackers   392 Jan 16  2010 bar.cc
-rw------- 1 alice hackers 56438 Oct  5  2007 core
drwxr-xr-x 2 alice hackers  4096 Apr 28 11:05 cs229/
drwxr-xr-x 2 alice hackers  4096 Dec  3  2011 cs252/
-rw-r----- 1 alice hackers   937 Jan 16  2010 foo.cc
-rw-r----- 1 alice hackers    88 May  4 09:43 hello.c
-rw-r----- 1 alice hackers   104 May  3 18:23 hello.h
-rw------- 1 alice hackers  3584 May  4 09:46 hello.o
drwxr-xr-x 2 alice hackers  4096 Apr 26 14:44 math168/
drwxr-xr-x 2 alice hackers  4096 Dec  1  2009 se101/
prompt$
```

mv: move files

Usage (just like cp):

2. `mv src1 src2 ...srcn Directory`
   - `src1` : an existing file or directory
   - ⋮
   - `srcn` : an existing file or directory
   - `Directory` : An existing directory
   - Moves each source item into `Directory`

```
prompt$
```

oving (2)

mv: move files

Usage (just like cp):

2. `mv src1 src2 ...srcn Directory`
   - `src1` : an existing file or directory
   - ⋮
   - `srcn` : an existing file or directory
   - `Directory` : An existing directory
   - Moves each source item into `Directory`

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

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)

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
```

cp: copy files

Usage:

2. `cp src1 src2 ...srcn Directory`
   - `src1` : an existing file
   - ⋮
   - `srcn` : 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

```
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

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

We can list out several files using *

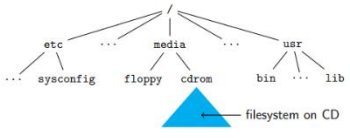
- Will generate a list of matching files
- Means, "fill in with zero or more characters"
- Usually does not match a leading .
- Can be used with any utility that takes lists of files

Examples with *

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

```
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$
```

Example: mounting and unmounting



1. Insert CD
2. Mount CD filesystem at /media/cdrom
3. Work with files ...
4. Close all files
5. Unmount CD filesystem
6. Eject CD

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

chmod ug+r foo.txt

Turns on read permission for user and group

chmod u=rw foo.txt

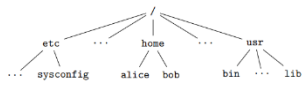
Sets user permissions to rw-

chmod a-x foo.txt

Turn off execute permission for user, group, and other

Filesystem management in *NIX

- There is only one system–wide tree of files



- Subdirectory separator is (forward) slash, e.g., /home/alice/.bashrc
- Logical drives are integrated into the system–wide tree
  - `Mount` : Add a filesystem (on a logical drive) to the tree
  - `Mount point` : Directory where a filesystem is mounted
  - `Unmount` : Remove a logical drive from the tree

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

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.

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$
```

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

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

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

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

FAT (File Allocation Table), Microsoft

- Common for DOS and Windows 9x machines

NTFS (New Technology File System), Microsoft

- Used for Windows NT, XP, and later
- Version 3.1 is current

HFS (Hierarchical File System), Apple

- HFS: 1985
- HFS+: 1998

APFS (Apple File System), Apple

- Released 2017
- Optimized for solid-state drive storage

ext (extended file system), Linux

- Started with ext (1992)
- ext4 (2008) is current

- `/bin` : Essential system binaries
- `/boot` : Kernel image and configuration
- `/dev` : Devices. More on these later.
- `/etc` : System configuration files
- `/home` : Directories for user files
- `/lib` : Software libraries (similar to dlls)
- `media` : Mount points for removable media
- `/tmp` : Temporary files, automatically deleted
- `/usr` : Most other system binaries
- `/var` : Variable, runtime stuff

```
prompt$ df -h
Filesystem      Size  Used  Avail  Use%  Mounted on
rootfs           34G  3.0G    31G    9%  /
devtmpfs        3.8G     0   3.8G    0%  /dev
tmpfs           3.8G     0   3.8G    0%  /dev/shm
tmpfs           3.8G  936K   3.8G    1%  /run
tmpfs           3.8G     0   3.8G    0%  /sys/fs/cgroup
tmpfs           3.8G     0   3.8G    0%  /media
/dev/sda3        48G   26G    23G   53%  /mnt/Win
/dev/sda2       355G   66G   290G   19%  /mnt/Mac
/dev/sda7        20G  824M    18G    5%  /home
/dev/sda5       497M   75M   398M   16%  /boot
/dev/sda4       128M  4.1M   124M    4%  /boot/efi
prompt$
```

Almost actual output, on a triple-boot Macintosh

- `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

su: substitute user

- More precisely: run a shell as another user
- Usage: `su [userid]`
- If no userid is specified, default is `root`
- You will be prompted for the user's password
  - Unless you are `root`

whoami: who am I

- Give the userid for the current shell
- The prompt may not change when you do `su`

Example /etc/fstab

```
/dev/mapper/vg_mbp13-lv01   /          ext4     defaults   1  1
/dev/sda5                   /boot      ext4     defaults   1  2
/dev/sda4                   /boot/efi  hfsplus  defaults   0  2
/dev/sda7                   /home      ext4     defaults   1  2
/dev/mapper/vg_mbp13-lv00   swap       swap     defaults   0  0
/dev/sda2                   /mnt/Mac   hfsplus  defaults   0  0
/dev/sda3                   /mnt/Win   ntfs     defaults   0  0
```

Globbing

- The shell allows arguments with "wildcards"
- Specifying wildcards:
  - `?` : fill in any one character
  - `*` : fill in any characters (zero or more)
  - `[list]` : fill in any character from the list
- The shell will replace the argument with matching path names
- If there are no matching path names:
  - Depends on the shell
  - Might give an error
  - Might leave the argument with the wildcard characters
- Fun fact
  - This is implemented in a C library function, `glob()`
  - See `man 3 glob` for more info
  - There are similar modules for other languages, e.g., Python

Bang (!) substitution

- `!!` : the previous command
- `!pre` : the most recent command starting with "pre"
- `!n` : the n^{th} command in the history
- `!$` : last argument from previous command
- `!*` : all arguments from previous command
- `!!:n` : n^{th} argument from previous command

You can use both command and argument selection:

- `!cat:$` : last arg. from last "cat" command
- `!402:3` : third argument from command 402 in history

Terminating, suspending, and resuming jobs

Foreground mode

- `Ctrl–c` : Terminate the foreground job
  - Usually works...
  - The job is interrupted, and destroyed
  - Memory is freed
  - We get the prompt back
- `Ctrl–z` : Suspend the foreground job
  - Usually works...
  - The job is interrupted, but not destroyed
  - In a GUI, window may not redraw itself
  - The job is still in memory
  - We can resume the job later if we want
  - We get the prompt back
- `%n` : Resume job *n* (in foreground mode)
  - The job number is given when you suspend it

wait: wait for jobs

Usage: `wait [%job] [%job] ...`

- Wait until *all* specified background jobs have finished
- If no jobs are specified, waits for all jobs
- `wait %n` is not quite the same as `fg %n`
  - `wait` just waits
  - `wait` does not connect terminal input to a job
  - `wait` does not resume a stopped job

jobs: display jobs for this shell

- Lists jobs by job number
- Gives status for each job

`%n` or `fg n` or `fg %n` : Run job *n* in the foreground

- Works for jobs that are suspended, or running in the background

`%n&` or `bg n` or `bg %n` : Run job *n* in the background

- Works for jobs that are suspended, or running in the background
  - But not needed if the job is already running in the background

kill: signal a job

Usage:

1. `kill -l`
   - Lists the available signals
2. `kill [-signal] %n`
   - Send signal to job *n*
   - Default signal is: `TERM`

- Note: `kill` can send any signal, not just `KILL`
- Yes, the choice of name is unfortunate
  - Many signals are not intended to stop a program

- `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

```
prompt$ ps
  PID TTY          TIME CMD
12017 pts/0    00:00:00 bash
12233 pts/0    00:00:00 ps
prompt$
```

- These are my processes, tied to this terminal
- Column `PID`: process ID
- Column `TTY`: which terminal
- Column `TIME`: total CPU time used so far
- Column `CMD`: the command

kill: send a signal

Usage:

1. `kill -l`: list signals
2. `kill [-signal] %n`: send signal to job `n`
3. `kill [-signal] pid`: send signal to process `pid`
   - Only if you own the process, or are `root`

wait: wait for a job or process

Usage: `wait [%job] [pid] ...`

- `wait` is a shell builtin
- Can only wait for a process that is a child of the shell

Fun with >&

`cmd > out.txt 2>&1`

Both `stdout` and `stderr` go to `out.txt`

`cmd 2>&1 > out.txt`

`stdout` goes to `out.txt`, `stderr` goes to terminal

`cmd > outA.txt 2>&1 > outB.txt`

`stdout` goes to `outB.txt`, `stderr` goes to `outA.txt`

`cmd 2> outA.txt 1>&2 2> outB.txt`

`stdout` goes to `outA.txt`, `stderr` goes to `outB.txt`

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$
```

ComS 252 — Iowa State University

Misc. Processes Utilities Services Exit status Redirection Pipes

Process priority

- Every process has a *priority*
  - Integer value, influences the scheduler
  - Linux: higher integer means lower priority

nice: run a command with lower priority

Usage: `nice cmd arg1 arg2 ...`

renice: adjust the priority of processes

- Usage: `renice change pid ...`
- Ordinary user: can only lower priority, if you own the process
- `root`: can raise or lower priority of any process

Answers to motivating questions

1. How does a job know where and how to display its output?
   Actually, it doesn't know. It just writes to `stdout` and `stderr`.
2. When I run multiple jobs, why don't they clobber each other?
   Each job becomes a process with its own memory space.
3. What happens to a job if its shell terminates?
   It depends. Typically, stopped jobs will terminate, and running background jobs will keep running.
4. How can I control a job from another shell?
   Find its process ID and control the process directly.
5. Can I tell if a job finished successfully?
   Yes, by checking the exit status. This can only be done in the same shell.

Summary of today's commands

- `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?

`df` : show disk space available on devices
`fdisk` : disk partitioning
`ln` : link files
`mkfs` : create a filesystem ("format a disk")
`mount` : mount a filesystem
`touch` : change file time
`umount` : un-mount a filesystem

`/dev/hda` : First IDE drive
`/dev/hdb` : Second IDE drive
`/dev/hdc` : Third IDE drive
`/dev/hdd` : Fourth IDE drive
`/dev/sda` : First SATA or SCSI drive
`/dev/sdb` : Second SATA or SCSI drive
⋮

To access a particular partition, append the partition number; e.g.,
`/dev/hda1` : First IDE drive, partition 1
`/dev/hdc5` : Third IDE drive, partition 5
`/dev/sdb4` : Second SATA or SCSI drive, partition 4

Formatting and such

mkfs

- Builds a filesystem ("formats a disk")
- You need to specify the device (usually, a partition)
  - Or you can specify a file and a size to build a "disk image"
  - But: use `mkisofs` instead, to build an ISO image

`-t` : Specify the filesystem type (or get the default)

```
prompt# mkfs -t vfat /dev/hdb3
```

touch: change file time

- Usage: `touch file1 ... filen`
- Default: sets modification and access times to "now"
- Will create empty files for any that do not exist
- Can set an arbitrary[1] time in the future or past
- Check the man page for more details

Partitioning tools

fdisk

- Just like the old MS-DOS utility
- Not the easiest to use, but gets the job done
- Need to specify the device to partition:

```
prompt# fdisk /dev/hdb
```

parted

- GNU partition editor; "smarter" than `fdisk`
- Can handle GUID partition tables
- If you don't specify a device, it will guess

Mounting and unmounting filesystems

mount: Add a device to the filesystem tree

- Specify the device, and the mount point

`-t` : Specify the filesystem type (`mount` can sometimes guess)

```
prompt# mount -t vfat /dev/hdb2 /Win/D
```

umount: remove a device from the filesystem tree

- Only need to specify the mount point, or the device
- Will fail if the device is busy

```
prompt# umount /Win/D
```

Disk usage

df

- For each mounted filesystem, shows
  - The space used
  - The space available

`-h` : Use "human readable" sizes
- E.g., "1.2G" instead of "1288490189"

A file has 3 times associated with it

1. Modification time: when its contents changed
2. Status time: when its group, owner, permissions changed
3. Access time: when it was last read

- `ls -l`: gives modification time by default
- `ls -cl`: show status time
- `ls -ul`: show access time

Suppose a disk reports itself to contain 1024 cylinders, 16 heads, and 30 sectors per track. Assuming 512-byte sectors, what is the capacity of this disk?

× 0

Correct Answer: 251 Megabytes, 251 Mb, 240 Mebibytes, 240 Mib

Capacity = 251,658,240 bytes

1 byte = 8 bits, so we need to convert bytes to bits:

Capacity = 251,658,240 bytes x 8 bits/byte = 2,013,265,920 bits

Capacity = 251,658,240 bytes ÷ (2^20 bytes/MiB) = 240 MiB (rounded to the nearest whole number).

6 0 / 1 point

Give a Linux command to run a utility named myutil, except everything read from standard input is obtained from device /dev/microphone.

user$ × ls

Correct Answer: myutil < /dev/microphone

7 0 / 1 point

Give a Linux command to display the items contained in directory /etc. Assume the current working directory is /home/alice.

alice$ × ls

Correct Answer: ls /etc, cd /etc; ls

8 0 / 1 point

Give a single Linux command to remove all files in the current working directory whose filename ends in .o

user$ × dr.o

Correct Answer: rm *.o

9 0 / 1 point

Give a Linux command, or sequence of commands, to rename the file named THIS NAME HAS SPACES.doc in the current working directory. The new name should be better_name.doc.

user$ × 0

Correct Answer: mv "THIS NAME HAS SPACES.doc" better_name.doc, mv THIS\ FILE\ HAS\ SPACES.doc better_name.doc

10 0 / 1 point

Suppose we have just moved a configuration file named cupsd.conf into the current working directory, and it has the following long listing:

```
-rwxrwxrwx  1 nobody nobody 22082 Feb  9 2010  cupsd.conf
```

Give a sequence of Linux commands (separated by ;) to execute, as root, which will modify the long listing to become:

```
-rw-r-----  1 root   lp     22082 Feb  9 2010  cupsd.conf
```

You may assume that the group lp already exists.

root$ × 0

Correct Answer: chmod 640 cupsd.conf; chown root:lp cupsd.conf, chmod u=rw cupsd.conf; chmod g=r cupsd.conf; chmod o= cupsd.conf; chown root:lp cupsd.conf

11 0 / 1 point

Suppose we have just executed the following, in a Linux shell:

```
alice$ myapp &
[1]    5577
alice$
```

Give two different ways to terminate this myapp job, from the shell.

Method 1: × 0

Correct Answer: kill %1

Method 2: × 0

Correct Answer: fg %1 then Ctrl-C

12 0 / 1 point

Suppose you have temporarily connected a second SATA drive to your computer. The third partition on this disk is an ext4 filesystem. Give the Linux command, executed as root, to mount this partition to the (existing) mount point /oldhome.

root$ × 0

Correct Answer: mount /dev/sdb3 /oldhome

13 0 / 1 point

Suppose directory /oldhome contains users' home directories from an old system. Give a Linux command or series of commands, executed as alice, to make alice's current home directory identical to her old home directory /oldhome/alice, assuming that directory /home is currently empty.

alice$ × 0

Correct Answer: cp -Rp /oldhome/alice /home, cd /home; cp -Rp /oldhome/alice ., cp -R -p /oldhome/alice /home

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

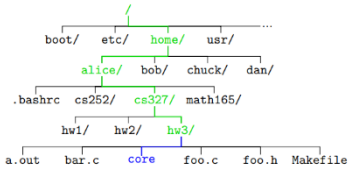
- Important "small" prefixes in computing:

Prefix	Sym.	Multiplier
milli	m	10^{-3}
micro	μ	10^{-6}
nano	n	10^{-9}
pico	p	10^{-12}

- Important "large" prefixes in computing:

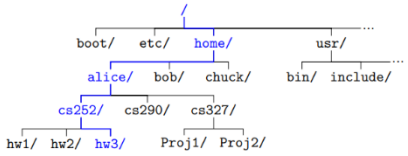
Prefix	Sym.	Multiplier
kilo	k	10^{3}
mega	M	10^{6}
giga	G	10^{9}
tera	T	10^{12}
peta	P	10^{15}
exa	E	10^{18}
zetta	Z	10^{21}

- Note: most symbols are capital letters
- These are used for capacities, and frequencies
  - 1 Tb drive (1 Terabyte: 1 trillion bytes)
  - 2 Ghz clock (2 Gigahertz: 2 billion cycles per second)
- 1984: US Department of Justice splits up the Bell System
- AT&T can legally start selling UNIX
- AT&T moves to commercialize UNIX
  - New licenses not as favorable for academic use
  - Berkeley begins removing AT&T code
- 1992: Unix System Laboratories sues Berkeley Software Design (loses)
- 1994: AT&T sells all rights to UNIX to Novell
- 1995: Novell "partially" sells to SCO
- 2000: SCO sells to The SCO Group
- 2003 and after: The SCO Group sues everybody about Linux



Absolute path: /home/alice/cs327/hw3/core
Working directory /home/alice/cs327/hw3
Relative pathname core

- . : current directory
  - Useful for relative paths
  - May appear in absolute paths
- .. : parent directory ("up one")
  - Useful for relative paths
  - May appear in absolute paths
  - In root directory, acts like "."
- ~ : current user's home directory
  - Only valid at the start of a path
  - Expanded by the shell
- ~user : another user's home directory
  - Only valid at the start of a path
  - Expanded by the shell




```
prompt$ pwd
/home/alice
prompt$ cd cs252/hw3
prompt$ cd ../hw2
```

mkdir: make (empty) directories

- Arguments: (absolute or relative) pathnames

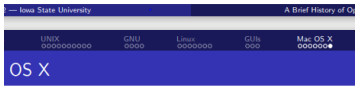
-p : will create intermediate directories if necessary

rmdir: remove empty directories

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

-p : will remove parent directories also

- 1983: Jobs hires Pepsi-Cola CEO, John Sculley
- 1985: Steve Wozniac leaves Apple
- 1985: Jobs demoted from head of Macintosh division
- 1985: Jobs quits Apple, founds "NeXT"
- 1988: NeXT workstation computer introduced
  - Target is University use
  - Competes with Sun
- NeXT machines run NeXTSTEP OS
  - "Unix–like"
  - Based on Mach kernel plus BSD
- Fun fact: WWW was invented on a NeXT Computer



- 1993: John Sculley leaves Apple
- 1996: Apple buys NeXT
  - Mac OS 9 needs to be updated
  - Want to base replacement on NeXTSTEP OS
- 1997: Jobs returns to Apple
- 2001: Mac OS X released
  - Still "Unix–like"
  - Uses Quartz for GUI
  - An implementation of X is available for compatibility

- POSIX: Portable Operating System Interface
- A family of standards for compatibility between OS's
  - Standards are specified by the IEEE
- Goal is to keep UNIX space coherent

100% POSIX compliant:
- BSD
- HP-UX
- Mac OS X

Mostly POSIX compliant
- FreeBSD, NetBSD, OpenBSD
- GNU/Linux

/ . refers to the root directory.

Just ls is short

- Ken Thompson wants a "personal" copy of MULTIC
  - Uses a discarded PDP-7 minicomputer
  - Ports 'Space Travel" game with Dennis Ritchie
  - Writes a stripped–down version of MULTICS
    - In assembly language
    - In about a month
  - Develops notion of a *process*

- B is too weak to implement UNIX
- Ritchie designs successor(s) to B, named "C"
  - Brian Kernighan and Dennis Ritchie write the first book on C
  - ANSI standardizes C in 1989, 1990, 1999, and 2011
  - C is still the language of choice for system programming
- Thompson and Ritchie successfully rewrite UNIX in C
- Other features added to UNIX

1974 – Thompson and Ritchie publish landmark UNIX

1991 –Linux started by Linus Torvalds, has built in kernel

- Steve Wozniak and Steve Jobs released Apple I in 1976
  - First computer consisting of a single circuit board
- Apple II released 1977
  - First personal computer with color graphics
- These were basic machines with a limited OS

- In computing, it is often easier to group 2^n things
  - E.g., $2^{10} = 1024$ bytes is more natural than 1000 bytes
- Slightly different prefixes used:

Confusing	Preferred	Sym.	Multiplier
kilo	Kibi	Ki	$2^{10} = 1024^1$
mega	Mebi	Mi	$2^{20} = 1024^2$
giga	Gibi	Gi	$2^{30} = 1024^3$
	Tebi	Ti	$2^{40} = 1024^4$

Moore's Law

Predicts that the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately ever two years

- Not a *physical* law; rather, a "rule of thumb"
- Moore made the original prediction in 1965
  - Originally, doubled *every year*
  - Moore predicted it would hold for 10 years
  - In 1975, Moore altered it to *every two years*
  - Industry has treated Moore's law as a target
- Is the end near for Moore's law?
  - 2017: 5 nm chips announced (~ 30 billion transistors)
  - 2021: 2 nm chips announced (~ 50 billion transistors)
    - Human DNA strand is 2.5 nm in diameter
- Address register size determines amount of addressable RAM
  - 20 bits (old 8086): 2^{20} bytes addressable = 1 Mb
  - 32 bits: 2^{32} bytes = 4 Gibi
    - 4 Gb limit on 32–bit machines
  - 48 bits: 2^{48} bytes = 256 Tebi
  - 64 bits: 2^{64} bytes = 16 Exabytes (16 billion Gb)
- Most systems have switched to 64–bit
  - Not straightforward — need 64–bit OS
  - Similar to change from 16 to 32 bits in early 1990's

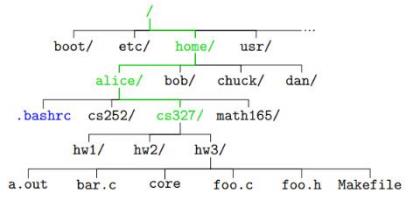
-S : Specify a list of sections to check
- Default is to check all sections

-a : Print all matching pages (e.g., in multiple sections)
- Default: only the first matching page is shown

-k : Check all pages for keywords
- Use this when you don't know which command

E.g., want to know something about a floppy drive:

```
prompt$ man -k floppy
fdformat (8)    - low-level format a floppy disk
floppy (8)      - format floppy disks
mbadblocks(1)   - tests a floppy disk, and marks the bad
mformat (1)     - add an MSDOS filesystem to a low-level
prompt$
```



Some pathnames for user bob in w.d. /home/alice/cs327

1. /home/alice/.bashrc
2. ~/../alice/.bashrc
3. ~alice/.bashrc
4. ../cs252/../.bashrc

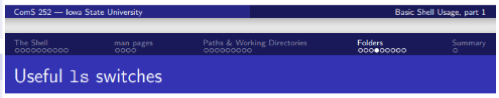
- cd : Change working directory.
- info : Fancy browsing of the online manual.
- ls : List the contents of a directory.
- man : Browse the online manual.
- mkdir : Create a directory.
- pwd : Print working directory.
- rmdir : Remove a directory.
- type : Is a command built-in, or not?

Listing contents

ls: list files (and folders)

- Without arguments: show files in working directory
- With arguments
  - File: show listing for the file
  - Directory: show files in the directory

```
prompt$ pwd
/home/alice
prompt$ ls
a.out     core     cs252    hello.c  hello.o  se101
bar.cc    cs229    foo.cc   hello.h  math168
prompt$ ls cs252
hw1    hw2    hw3    hw4
prompt$
```



-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)

Richard Stallman

- Annoyed by software licenses
- 1984: Quits MIT AI Lab, starts "GNU" project
  - GNU: GNU's Not UNIX (acronym with recursive definition)
- Goal of GNU: develop a complete UNIX–like OS but *avoid licensing issues*
- Wrote first versions of
  - gcc: C compiler
  - gdb: debugger
  - Emacs: text editor
- See http://www.gnu.org
- 1985: founds *Free Software Foundation*

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