Today

- Unix as an OS case study
- Intro to Shell Scripting

Login
Open your favorite text editor
Open a terminal

Review

- What is an Operating System?
- What are its goals?

Review: OS Goals

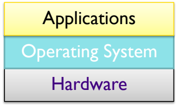
- Make computers easier to use
  - Abstraction!
  - Bridge gap between hardware and user experience
- Use computer hardware efficiently

Why are these two separate goals?

What is a "computer"?

Review: What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
  - Resource allocator
  - Control program
- Tasks:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner



What is an Operating System?

- Formally: A program that acts as an intermediary between the computer user and the computer hardware
- Goals:
  - Make the computer system easy to use.
  - Use the computer hardware efficiently.
- It is an extended machine
  - Hides the messy details which must be performed
  - Presents user with a virtual machine, easier to use
- It is a resource manager
  - Each program gets time with the resource
  - Each program gets space on the resource

SYSTEMS PROGRAMMING

What is Systems Programming?

- Program development with system tools (no fancy pants IDEs here)
- Uses system calls that hook in to core OS functions
- Use coding standards to ensure portability
  - Common file locations
  - Common compilation & installation procedures
  - Basic shell functionality
- We'll be programming in the Unix environment, using C

Why Unix?

- Open source = easier to study
  - Windows is proprietary & closed
  - OSX is proprietary and is built on top of Unix
- Historic: developed in the 60s & 70s
  - One of the oldest OS's in use today
- Most serious programmers & hackers know their way around Unix/Linux
- Linux is a Unix-like OS

Why Unix?

MY BOSS SAYS WE NEED SOME EUNUCH PROGRAMMERS.
I THINK HE MEANS UNIX, NOT EUNUCHS. AND I ALREADY KNOW UNIX.
IF THE COMPANY NURSE DROPS BY, TELL HER I SAID, "NEVER MIND."
S. Adams
E-Mail: SCOTTADAMS@AOL.COM
© 1993 United Feature Syndicate, Inc.

Unix Philosophy

- Make each program do one thing well
  - More complex functionality by combining programs
  - Make every program a filter
  - More efficient
  - Better for reuse
- Portability
- No GUIs
- Only error feedback

Why C?

- The high-level language (HLL) that's closest to the hardware

The System Programmer's Toolbox

- Shell: a program used to run other programs
- Text editor: where you'll develop your code
  - Your faves?
- Compiler: transforms source code into an executable file
  - gcc
- Debugger: a program that allows you to step through an execution & observe how the program state (i.e., variable values) changes
  - gdb
  - Print statements

What is a Shell?

- User interface to the operating system
- Command-line ***interpreter***
- Functionality:
  - Execute other programs
  - Manage files
  - Manage processes
- A program like any other
- Basic form of shell:

```
while <read command>:
   parse command
   execute command
```

hides details of underlying operating system

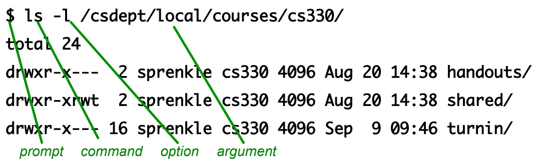
Unix Commands

- Each command performs (variations of) a single task
  - *options* can be used to modify what a command does
  - Multiple commands can be "piped" together to perform more complex tasks
- Syntax:
  `command options arguments`
- Examples:

Command	Options	Arguments
pwd		
cd		~/public_html
ls	-a -l	
ls	-al	~/public_html

Options can (usually) be combined together: these are equivalent

Example of `ls` Command



```
$ ls -l /csdept/local/courses/cs330/
total 24
drwxr-x---   2 sprenkle cs330 4096 Aug 20 14:38 handouts/
drwxr-xrwt   2 sprenkle cs330 4096 Aug 20 14:38 shared/
drwxr-x---  16 sprenkle cs330 4096 Sep  9 09:46 turnin/
```

prompt *command* *option* *argument*

Unix Commands

- Each command performs (variations of) a single task
  - *options* can be used to modify what a command does
  - Multiple commands can be "piped" together to perform more complex tasks
- Syntax:
  `command options arguments`
- Examples:

Not always required: may have default values

Command	Options	Arguments
pwd		
cd		~/public_html
ls	-a -l	
ls	-al	~/public_html

Default value is current directory

Examples of Unix commands

- Figuring out one's current directory: **pwd**
- Moving to another directory: **cd *targetdir***

Examples:

cd /	move to the root of the file system
cd ~ (also: just "**cd**" by itself)	move to one's home directory
cd /usr/local/src	move to /usr/local/src
cd ../..	move up two levels

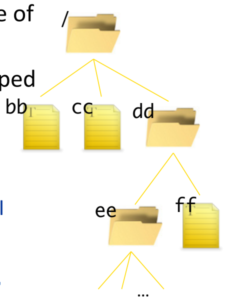
Program Development Process

- Divide & conquer: break the big programming problem into smaller subproblems
  - Recursively repeat as necessary
- Solve each subproblem & test for correctness
- In general, test your code after **every** change to catch bugs quickly & fix them easily
- Develop incrementally
- As the programs get bigger, periodically save working versions (script or version control)

WORKING WITH UNIX FILES

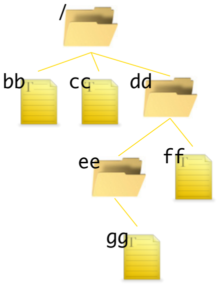
The file system

- A ***file*** is basically a sequence of bytes
- Collections of files are grouped into ***directories*** (≈ folders)
- A directory is itself a file
  - file system has a hierarchical structure (i.e., like a tree)
  - the root is referred to as "/"



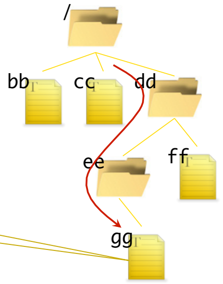
Referring to files: Absolute Paths

- An ***absolute path*** specifies how to get to a file starting at the file system root
  - List the directories on the path from the root /, separated by /



Referring to Files: Absolute Paths

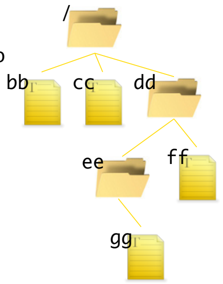
- An ***absolute path*** specifies how to get to a file starting at the file system root
  - List the directories on the path from the root /, separated by /



absolute path:
`/dd/ee/gg`

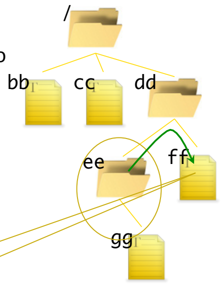
Referring to Files: Relative Paths

- Typically we have a notion of a ***current directory***
- A ***relative path*** specifies how to get to a file starting from the current directory
  - `..` means move up one level
  - `.` means current directory
  - list the directories on the path separated by /

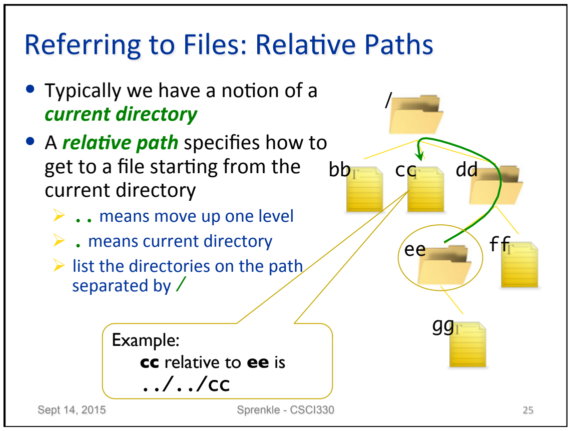


Referring to Files: Relative Paths

- Typically we have a notion of a ***current directory***
- A ***relative path*** specifies how to get to a file starting from the current directory
  - `..` means move up one level
  - `.` means current directory
  - list the directories on the path separated by /



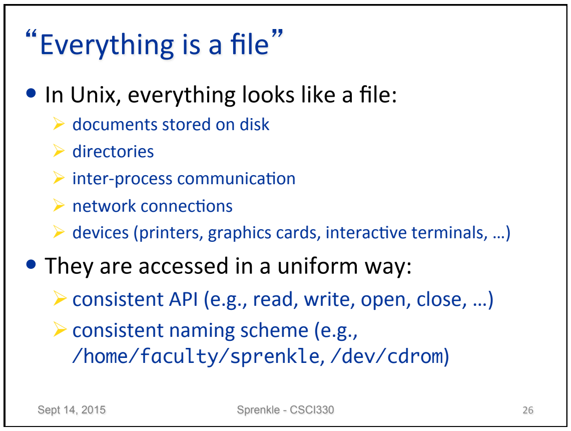
Example:
ff relative to **ee** is
`../ff`



Referring to Files: Relative Paths

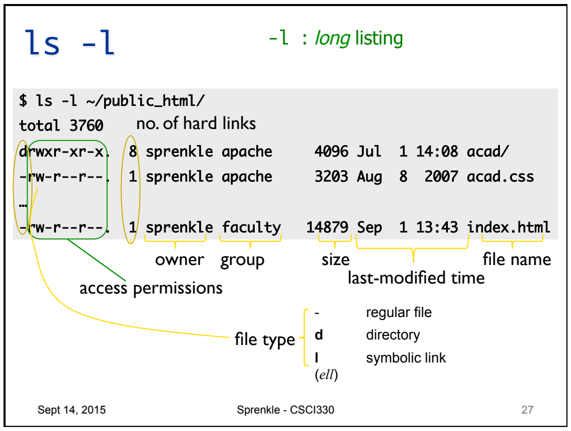
- Typically we have a notion of a ***current directory***
- A ***relative path*** specifies how to get to a file starting from the current directory
  - `..` means move up one level
  - `.` means current directory
  - list the directories on the path separated by `/`

Example:
cc relative to **ee** is
`../../cc`



"Everything is a file"

- In Unix, everything looks like a file:
  - documents stored on disk
  - directories
  - inter-process communication
  - network connections
  - devices (printers, graphics cards, interactive terminals, …)
- They are accessed in a uniform way:
  - consistent API (e.g., read, write, open, close, …)
  - consistent naming scheme (e.g., `/home/faculty/sprenkle`, `/dev/cdrom`)

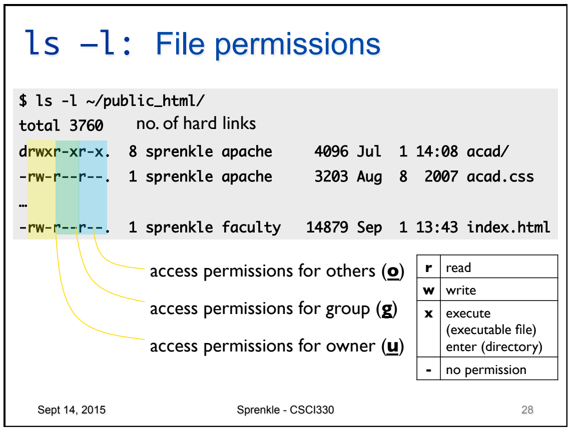


`ls -l`

`-l` : *long* listing

```
$ ls -l ~/public_html/
total 3760
drwxr-xr-x.  8 sprenkle apache     4096 Jul  1 14:08 acad/
-rw-r--r--.  1 sprenkle apache     3203 Aug  8  2007 acad.css
…
-rw-r--r--.  1 sprenkle faculty   14879 Sep  1 13:43 index.html
```

-	regular file
d	directory
l (*ell*)	symbolic link

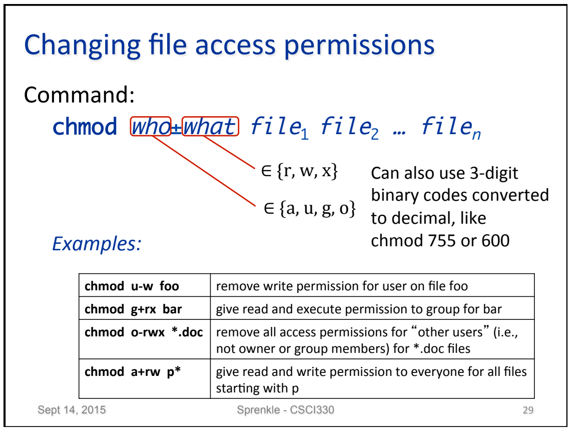


`ls –l:` File permissions

```
$ ls -l ~/public_html/
total 3760
drwxr-xr-x.  8 sprenkle apache     4096 Jul  1 14:08 acad/
-rw-r--r--.  1 sprenkle apache     3203 Aug  8  2007 acad.css
…
-rw-r--r--.  1 sprenkle faculty   14879 Sep  1 13:43 index.html
```

no. of hard links
access permissions for others (o)
access permissions for group (g)
access permissions for owner (u)

r	read
w	write
x	execute (executable file) enter (directory)
-	no permission



Changing file access permissions

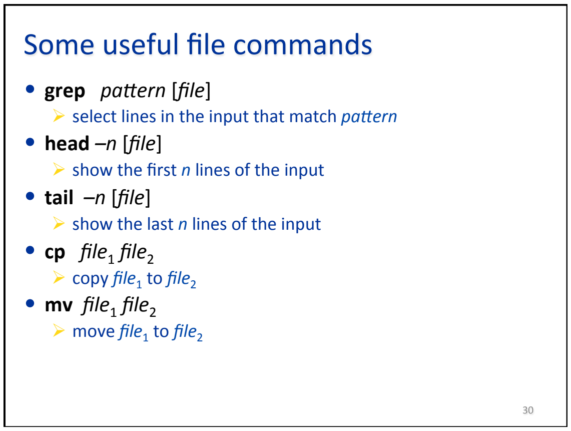
Command:

chmod *who*±*what* *file*$_1$ *file*$_2$ … *file*$_n$

Can also use 3-digit binary codes converted to decimal, like chmod 755 or 600

Examples:

chmod u-w foo	remove write permission for user on file foo
chmod g+rx bar	give read and execute permission to group for bar
chmod o-rwx *.doc	remove all access permissions for "other users" (i.e., not owner or group members) for *.doc files
chmod a+rw p*	give read and write permission to everyone for all files starting with p



Some useful file commands

- **grep** *pattern* [*file*]
  - select lines in the input that match *pattern*
- **head** –*n* [*file*]
  - show the first *n* lines of the input
- **tail** –*n* [*file*]
  - show the last *n* lines of the input
- **cp** *file*$_1$ *file*$_2$
  - copy *file*$_1$ to *file*$_2$
- **mv** *file*$_1$ *file*$_2$
  - move *file*$_1$ to *file*$_2$

Other File-Related Commands

Command	Purpose
`file`	Determine file type
`basename`	Strip directory and suffix from file names
`dirname`	Strip non-directory suffix from file name
`wc`	Print number of newlines, words, and bytes in files `-l` : lines `-m` : chars `-w` : words

LEARNING NEW UNIX COMMANDS

Finding out about commands

Figuring out which command to use

apropos *keyword*

man –k *keyword*

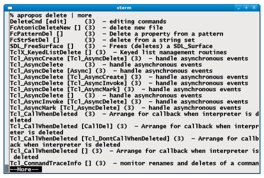
"searches a set of database files containing short descriptions of system commands for keywords"

- Helpful, but not a panacea:
  - depends on appropriate choice of keywords
    - may require trial and error
  - may return a lot of results to sift through
    - pipe through **more** or **less**
- You can also use Google & stackoverflow

Example: Deleting a file

Figuring out which command to use:

- **apropos delete**
  - produces many screenfuls of output that go by too quickly
- **apropos delete | more**
  - many screenfuls of output, but shown one screenful at a time
  - most of the commands shown aren't relevant



```
% apropos delete | more
DeleteCmd [edit]      (3)  - editing commands
FcAtomicDeleteNew [] (3)  - delete new file
FcPatternDel []       (3)  - Delete a property from a pattern
FcStrSetDel []        (3)  - delete from a string set
SDL_FreeSurface []    (3)  - Frees (deletes) a SDL_Surface
TclX_KeyedListDelete [] (3)  - Keyed list management routines
Tcl_AsyncCreate [Tcl_AsyncDelete] (3)  - handle asynchronous events
Tcl_AsyncDelete       (3)  - handle asynchronous events
Tcl_AsyncDelete [Async] (3)  - handle asynchronous events
Tcl_AsyncDelete [Tcl_AsyncCreate] (3)  - handle asynchronous events
Tcl_AsyncDelete [Tcl_AsyncInvoke] (3)  - handle asynchronous events
Tcl_AsyncDelete [Tcl_AsyncMark] (3)  - handle asynchronous events
Tcl_AsyncDelete []    (3)  - handle asynchronous events
Tcl_AsyncInvoke [Tcl_AsyncDelete] (3)  - handle asynchronous events
Tcl_AsyncMark [Tcl_AsyncDelete] (3)  - handle asynchronous events
Tcl_CallWhenDeleted  (3)  - Arrange for callback when interpreter is d
eleted
Tcl_CallWhenDeleted [CallDel] (3)  - Arrange for callback when interpr
eter is deleted
Tcl_CallWhenDeleted [Tcl_DontCallWhenDeleted] (3)  - Arrange for callb
ack when interpreter is deleted
Tcl_CallWhenDeleted [] (3)  - Arrange for callback when interpreter is
 deleted
Tcl_CommandTraceInfo [] (3)  - monitor renames and deletes of a comman
--More--
```

Finding out about commands

Figuring out how to use a command

man *command*

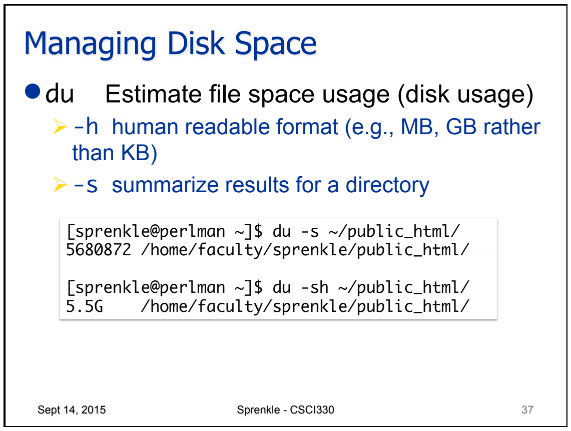
"displays the on-line manual pages"

- Provides information about command options, arguments, return values, bugs, etc.

Managing Disk Space

Command	Purpose	Options
du	estimate file space usage	`-h` human readable `-s` summarize
df	report filesystem disk space usage	`-h` human readable

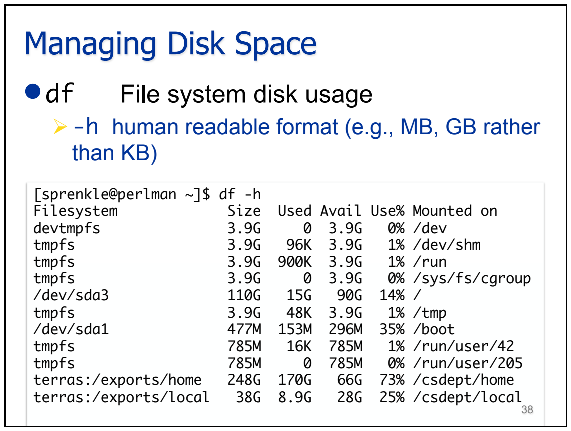
Many more options…
See man page



Managing Disk Space

- du   Estimate file space usage (disk usage)
  - -h human readable format (e.g., MB, GB rather than KB)
  - -s summarize results for a directory

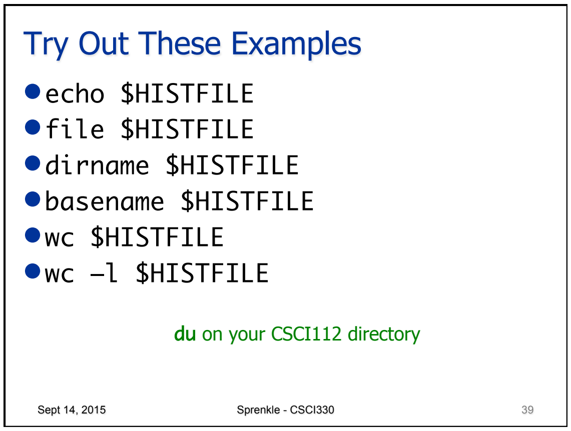
```
[sprenkle@perlman ~]$ du -s ~/public_html/
5680872 /home/faculty/sprenkle/public_html/

[sprenkle@perlman ~]$ du -sh ~/public_html/
5.5G    /home/faculty/sprenkle/public_html/
```



Managing Disk Space

- df   File system disk usage
  - -h human readable format (e.g., MB, GB rather than KB)

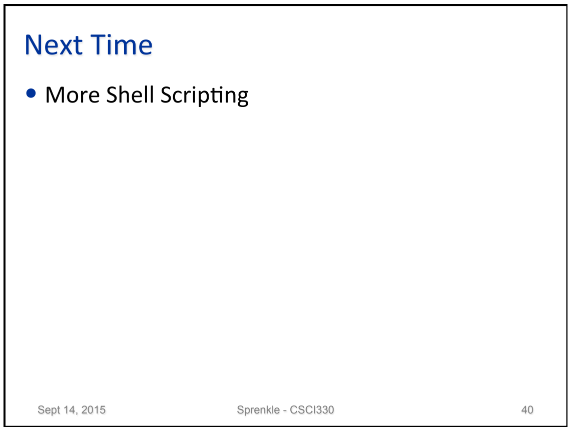
```
[sprenkle@perlman ~]$ df -h
Filesystem             Size  Used Avail Use% Mounted on
devtmpfs               3.9G     0  3.9G   0% /dev
tmpfs                  3.9G   96K  3.9G   1% /dev/shm
tmpfs                  3.9G  900K  3.9G   1% /run
tmpfs                  3.9G     0  3.9G   0% /sys/fs/cgroup
/dev/sda3              110G   15G   90G  14% /
tmpfs                  3.9G   48K  3.9G   1% /tmp
/dev/sda1              477M  153M  296M  35% /boot
tmpfs                  785M   16K  785M   1% /run/user/42
tmpfs                  785M     0  785M   0% /run/user/205
terras:/exports/home   248G  170G   66G  73% /csdept/home
terras:/exports/local   38G  8.9G   28G  25% /csdept/local
```



Try Out These Examples

- echo $HISTFILE
- file $HISTFILE
- dirname $HISTFILE
- basename $HISTFILE
- wc $HISTFILE
- wc –l $HISTFILE

du on your CSCI112 directory



Next Time

- More Shell Scripting