CS 297: Tools for the Software Life Cycle and Beyond

Goal: Productivity

- Many available tools
  - UNIX & UNIX-like systems (e.g., Linux)
  - Open-source (Gnu, Apache, Eclipse)
  - Proprietary
  - Variety of purposes
- Know what (free) tools are available, what they do, how to use them

Goal: Automation

- Often have to do a task over and over again
  - Time-intensive to do by hand
  - Shortcuts aren't enough
- What we want
  - Tools to make tasks easier
  - Scripts to be able to repeat the tasks easier

Main Types of Tools

- Command-line
- Graphical/GUI interfaces

What are the benefits and limitations of each type of tool?

Command-Line Tools

- Benefits
  - Flexible--lots of options
  - After run once, can run again in same terminal using up arrow key or using !command
  - Tab-completion
  - Automation: Can be put into bash scripts and repeated
- Limitations
  - Requires knowing name of command
  - Requires knowing syntax of command, options
    - Easy to screw up!
  - Slower learning curve

GUIs

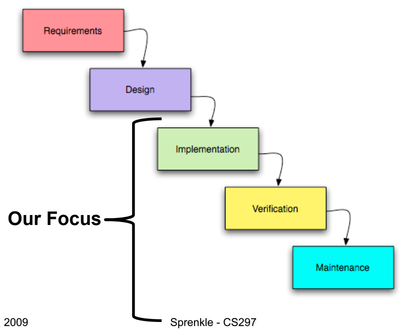
- Benefits
  - Require less knowledge of syntax
  - Generally: faster learning curve
- Limitations
  - Can require many clicks to do even simple operations
  - May require a lot of set up/configuration
  - Harder to automate, repeat tasks

Course Content

- Unix tools
- Bash scripting
- Software development tools

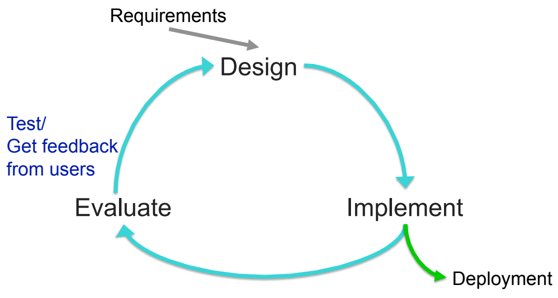
Apr 20, 2009 Sprenkle - CS297

Software Life Cycle: Waterfall Model



Apr 20, 2009 Sprenkle - CS297

Iterative Design



Apr 20, 2009 Sprenkle - CS297

Software Tools

- What are they?
- What is the goal of software tools?
- What is an IDE and its goal?

Apr 20, 2009 Sprenkle - CS297

Course Objectives

- At the end of this course, you will be able to
  - Use a variety of Unix tools
  - Apply a variety of tools to automate many tasks
  - Describe the use of state-of-the-art software tools for developing and maintaining large software systems, based on hands-on experience
  - Discuss when best to use different tools, the limitations of the tools, and what they have to offer
  - Discuss the challenges and strategies in building software tools
  - Communicate technical content in both oral and written forms

Apr 20, 2009 Sprenkle - CS297

Non-Syllabus Goals

- Improve your productivity
- Unix confidence/proficiency
  - To intermediate user
- Tool confidence
  - Less intimidated by installing, learning new tools
- Resume builder!
  - Impress potential employers, advisors
- Non-goal: System Administrator

Apr 20, 2009 Sprenkle - CS297

Expectations

- Material is most relevant in context
  - Need to make it relevant to you
  - What would you like to do--now or in the future?
  - What tools interest you?
- Actively explore tools
  - Try out everything we do
  - Make mistakes and learn from them

Grading

- (42%) Individual programming, reading, and homework assignments
- (15%) Midterm Exam
- (36%) Tool Demonstrations
- (7%) Professionalism: participation and attendance

UNIX

Our Heroes: UNIX Developers

Ken Thompson Dennis Ritchie

UNIX Philosophy

- Doug McIlroy, inventor of Unix *pipes*, a founder of Unix tradition:

  This is the Unix philosophy: Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface

- This is usually severely abridged to “do one thing and do it well”

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

The UNIX Philosophy

- Scripting increases leverage and portability

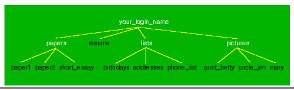
List the usernames of a system’s current users:

```
who | awk '{print $1}' | sort | uniq
```

We’ll talk more about piping on Wednesday…

The UNIX Philosophy

- Avoid captive interfaces
  - The user of a program isn’t always human
  - Look nice, but code is big and ugly
  - Problems with scale
- Silence is golden
  - Only report if something is wrong
- Think hierarchically



UNIX Highlights / Contributions

- Portability
  - Because implemented in C rather than assembly language (specific to machine), ran on variety of machines
- TCP/IP implementation -- 1984
  - Communicate btw different machines from different vendors
- Hierarchical file system; the file abstraction
- Multitasking and multiuser capability for minicomputer

UNIX Highlights / Contributions

- Inter-process communication
  - Pipes: output of one programmed fed into input of another
- Software tools
- Development tools
- Scripting languages

Quotes

- "Unix is simple. It just takes a genius to understand its simplicity." – Dennis Ritchie
- "UNIX was not designed to stop its users from doing stupid things, as that would also stop them from doing clever things." – Doug Gwyn
- "Unix never says 'please'." – Rob Pike
- "Unix is user-friendly. It just isn't promiscuous about which users it's friendly with." – Steven King
- "Those who don't understand UNIX are condemned to reinvent it, poorly." – Henry Spencer

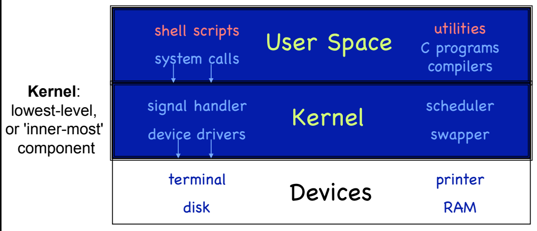
UNIX STRUCTURE

The Operating System

- The government of your computer
- Kernel: Performs critical system functions and interacts with the hardware
  - Loaded into memory during the boot process, and always stays in physical memory
  - Responsible for managing CPU and memory for processes, managing file systems, and interacting with devices
- Systems utilities: Programs and libraries that provide various functions through system calls to the kernel

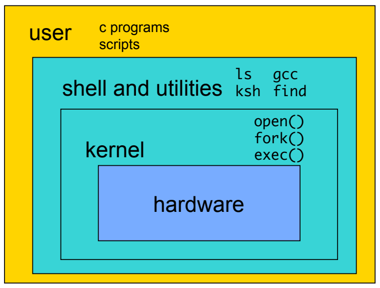
Apr 20, 2009 Sprenkle - CS297

UNIX Structural Layout



Apr 20, 2009 Sprenkle - CS297

UNIX System Structure



Apr 20, 2009 Sprenkle - CS297

What is a Shell?

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

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

hides details of underlying operating system

Apr 20, 2009 Sprenkle - CS297

Most Commonly Used Shells

● /bin/sh	The Bourne Shell / POSIX shell
● /bin/csh	C shell
● /bin/tcsh	Enhanced C Shell
● /bin/ksh	Korn shell
● /bin/bash	Free ksh clone

Which shell do we use in the lab?

Apr 20, 2009 Sprenkle - CS297

Shell Interactive Use

- When you open a terminal, you interactively use the shell:
  - Command history
  - Command line editing
  - File expansion (tab completion)
  - Command expansion
  - Key bindings
  - Job control

Apr 20, 2009 Sprenkle - CS297

Shell Scripting

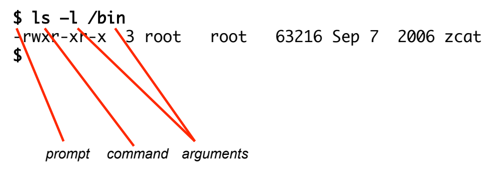
- A set of shell commands that constitute an executable program
- A shell script is a regular text file that contains shell or UNIX commands
- Very useful for automating repetitive tasks and administrative tools and for storing commands for later execution

More on this later…

Simple Commands

- Sequence of non-blank arguments separated by blanks or tabs
- 1st argument (numbered 0) usually specifies the name of the command to be executed
- Any remaining arguments:
  - Are passed as arguments to that command
  - Depending on command, arguments may be filenames, pathnames, directories or special options
  - Special characters are interpreted by shell

Example of Simple Command



```
$ ls -l /bin
-rwxr-xr-x  3 root   root   63216 Sep 7  2006 zcat
$
```

- Execute a basic command
- Parsing into command and arguments is called *splitting*

Types of Arguments

```
$ tar -c -v -f archive.tar main.c main.h
```

- Options/Flags
  - Convention: -*X* or --*longname*
- Parameters
  - May be files, may be strings
  - Depends on command

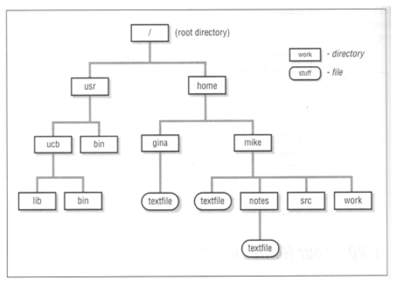
Basic Unix Tools

- File/Directory Management
- Process Management

Directory Management Review

- How is Unix's directory structure organized?

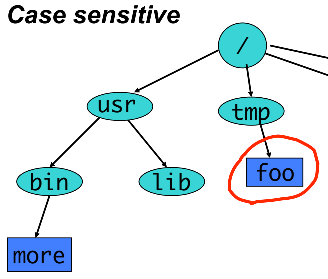
The UNIX File Hierarchy



Apr 20, 2009 Sprenkle - CS297

Definition: Filename

A sequence of characters other than slash
Case sensitive



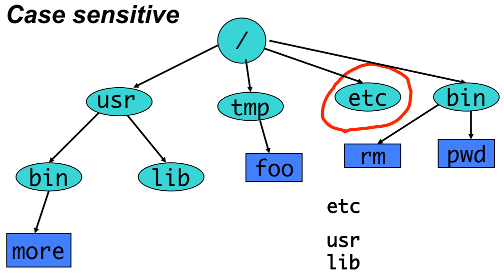
foo

rm
pwd
more

Apr 20, 2009 Sprenkle - CS297

Definition: Directory

Holds a set of files or other directories
Case sensitive



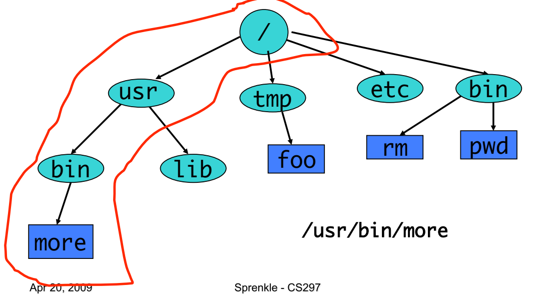
etc

usr
lib
bin

Apr 20, 2009 Sprenkle - CS297

Definition: Pathname

A sequence of directory names followed by a simple filename, each separated from the previous one by a /



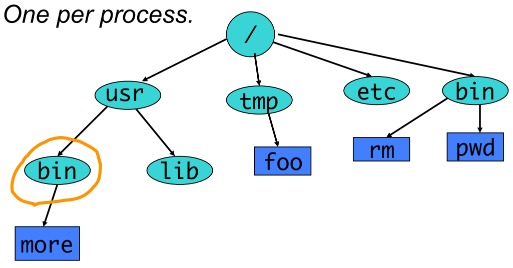
/usr/bin/more

Apr 20, 2009 Sprenkle - CS297

Definition: Working Directory

Directory the process is currently in.

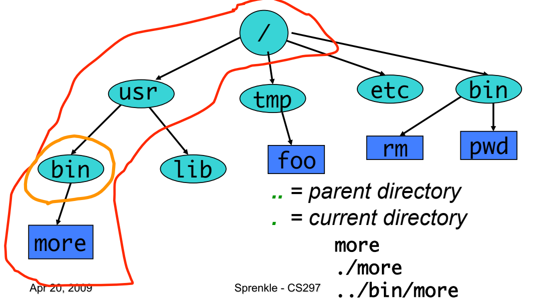
One per process.



Apr 20, 2009 Sprenkle - CS297

Definition: Relative Pathname

*A pathname **relative** to the working directory (as opposed to **absolute pathname**)*



.. = *parent directory*
. = *current directory*

```
more
./more
../bin/more
```

Apr 20, 2009 Sprenkle - CS297

Files and Directories

- Files are just a sequence of bytes
  - No file types (data vs. executable)
  - No sections
  - Example of UNIX philosophy
- Directories are a list of files and status of the files:
  - Creation date
  - Attributes
  - etc.

Apr 20, 2009 Sprenkle - CS297

Directory Management Review

- How do you see a directory's contents?
  - How can you find out more information about the contents?
  - How can you list the content in time order?
- How do you go into a directory?
  - Home directory?
  - Parent directory?
- How can you help avoid a lot of typing when you're trying to go into a directory?

Apr 20, 2009 Sprenkle - CS297

Tilde Expansion

- Each user has a *home* directory
- Most shells (ksh, csh) support ~ operator:
  - ~ expands to my home directory
    - `~/myfile` → `/home/kornj/myfile`
  - `~user` expands to user's home directory
    - `~unixtool/file2` → `/home/unixtool/file2`
- Useful because home directory locations vary by machine

What is your home directory?

Apr 20, 2009 Sprenkle - CS297

Directory Management Review

- How do you know what directory you're in?
- How do you make a new directory?
  - How do you make a series of directories, for example `cs297/practice/tmp`, in one command?
  - What if `cs297/practice/` doesn't exist?
- How do you delete an empty directory?

Apr 20, 2009 Sprenkle - CS297

File Management Review

- How do you copy a file?
  - A directory and its contents?
- How do you move/rename a file?
- What is the short cut for the current directory?
- How do you delete a file?
- How do you delete a whole directory?

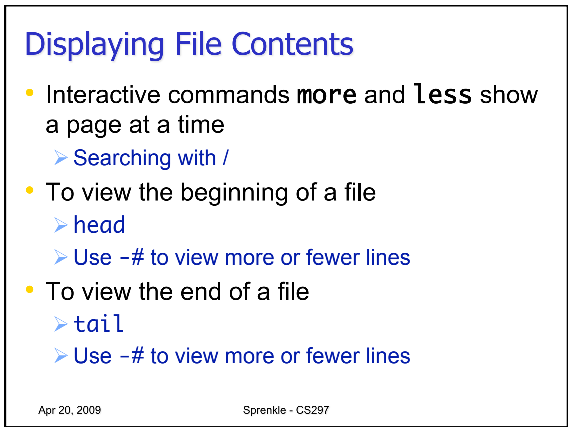
Apr 20, 2009 Sprenkle - CS297

Displaying File Contents

- `cat` can be used to display the contents of a file in the terminal
  - When invoked with a list of file names, it concatenates them
- Some options:
  - `-n` number output lines (starting from 1)
  - `-v` display control-characters in visible form (e.g. ^C)

Practice: handouts directory's last name file
→ Do **not** `cd` into that directory

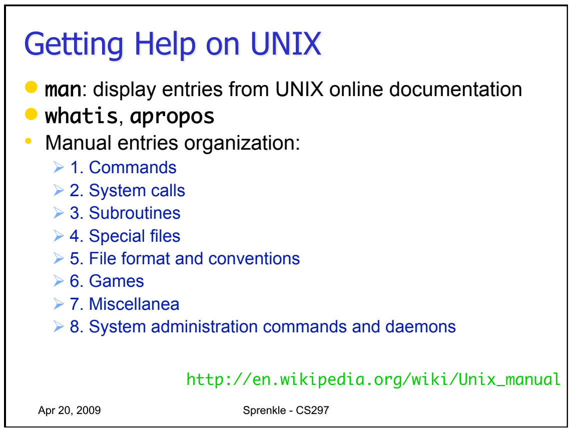
Apr 20, 2009 Sprenkle - CS297



Displaying File Contents

- Interactive commands `more` and `less` show a page at a time
  - Searching with /
- To view the beginning of a file
  - `head`
  - Use -# to view more or fewer lines
- To view the end of a file
  - `tail`
  - Use -# to view more or fewer lines

Apr 20, 2009 Sprenkle - CS297

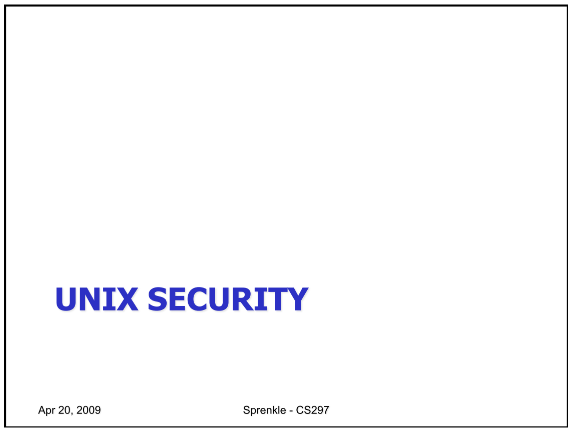


Getting Help on UNIX

- `man`: display entries from UNIX online documentation
- `whatis`, `apropos`
- Manual entries organization:
  - 1. Commands
  - 2. System calls
  - 3. Subroutines
  - 4. Special files
  - 5. File format and conventions
  - 6. Games
  - 7. Miscellanea
  - 8. System administration commands and daemons

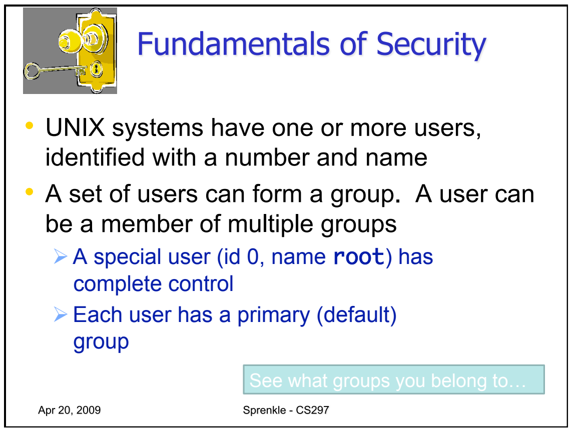
http://en.wikipedia.org/wiki/Unix_manual

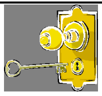
Apr 20, 2009 Sprenkle - CS297



UNIX SECURITY

Apr 20, 2009 Sprenkle - CS297



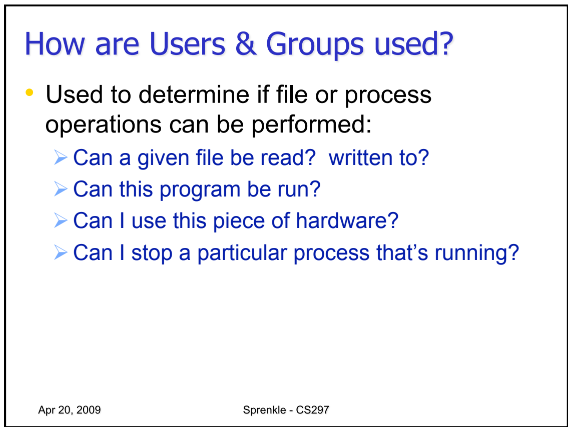


Fundamentals of Security

- UNIX systems have one or more users, identified with a number and name
- A set of users can form a group. A user can be a member of multiple groups
  - A special user (id 0, name `root`) has complete control
  - Each user has a primary (default) group

See what groups you belong to…

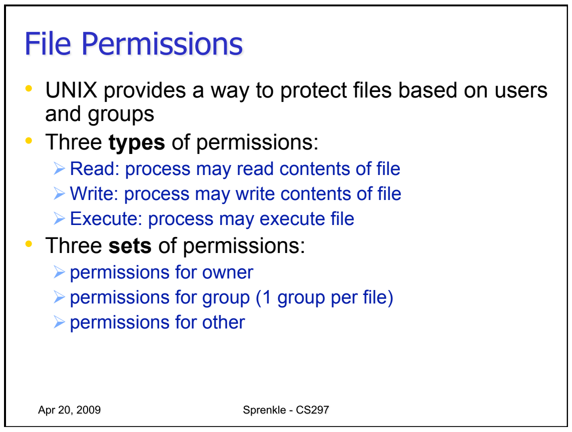
Apr 20, 2009 Sprenkle - CS297



How are Users & Groups used?

- Used to determine if file or process operations can be performed:
  - Can a given file be read? written to?
  - Can this program be run?
  - Can I use this piece of hardware?
  - Can I stop a particular process that's running?

Apr 20, 2009 Sprenkle - CS297



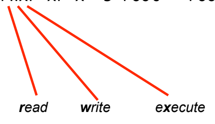
File Permissions

- UNIX provides a way to protect files based on users and groups
- Three **types** of permissions:
  - Read: process may read contents of file
  - Write: process may write contents of file
  - Execute: process may execute file
- Three **sets** of permissions:
  - permissions for owner
  - permissions for group (1 group per file)
  - permissions for other

Apr 20, 2009 Sprenkle - CS297

A simple example

```
$ ls -l /bin
-rwxr-xr-x  3 root   root   63216 Sep 7  2006 zcat
$
```



Directory permissions

- Same types and sets of permissions as for files:
  - **read**: process may read the directory *contents* (i.e., list files)
  - **write**: process may add/remove files in the directory
  - **execute**: process may open files in directory or subdirectories

Unix Permissions

- Categories: owner, group, others
- Permissions: read, write, execute

```
[sprenkle@hopper courses]$ ls -l /home/courses/cs209/handouts/
total 16
drwxr-x--- 3 sprenkle cs297   4096 2009-04-17 16:00 ./
drwxr-x--- 5 sprenkle cs297   4096 2009-04-15 16:20 ../
drwxr-xr-x 2 sprenkle faculty 4096 2009-04-17 12:57 day1/
-rw-r--r-- 1 sprenkle faculty    0 2009-04-17 16:00 tmp
permissions   owner   group   size   date modified   file name
```

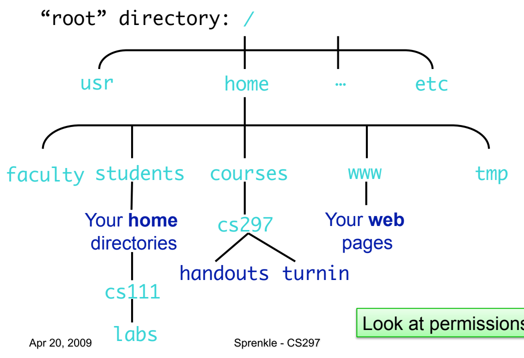
Unix Permissions

- Categories: owner, group, others
- Permissions: read, write, execute

```
[sprenkle@hopper courses]$ ls -l /home/courses/cs209/handouts/
total 16
drwxr-x--- 3 sprenkle cs297   4096 2009-04-17 16:00 ./
drwxr-x--- 5 sprenkle cs297   4096 2009-04-15 16:20 ../
drwxr-xr-x 2 sprenkle faculty 4096 2009-04-17 12:57 day1/
-rw-r--r-- 1 sprenkle faculty    0 2009-04-17 16:00 tmp
permissions   owner   group   size   date modified   file name
```

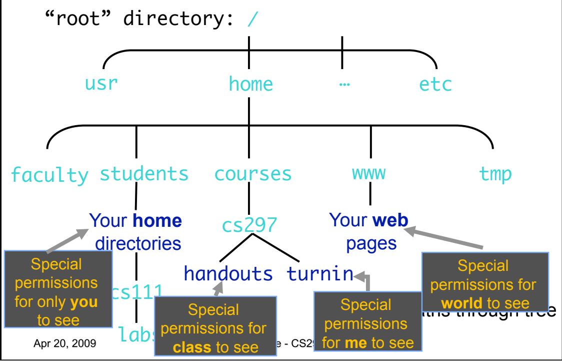
- What are the permissions on the file `tmp`?
- In the permissions, how can we distinguish between an executable file and directory?
- What does it mean for a file to be executable?

(Partial) Linux File System



Look at permissions

(Partial) Linux File System


"root" directory: /
usr
home
…
etc
faculty
students
courses
www
tmp
Your home directories
cs297
Your web pages
handouts
turnin
cs111
labs
Special permissions for only you to see
Special permissions for class to see
Special permissions for me to see
Special permissions for world to see

Utilities for Manipulating File Attributes

`chmod`	change file permissions
`chown`	change file owner
`chgrp`	change file group
`umask`	user file creation mode mask

- Only owner or super-user can change file attributes
- Upon creation, default permissions given to file modified by process's `umask` value

Changing Permissions

- `chmod` command
  - Syntax: `chmod [options] <mode> <file(s)>`
- Examples:

```
chmod u+x script.sh
chmod a-w readDir
chmod -R ug+r myDir
```

Recursive

Shorthand	Meaning
u	User/owner
g	Group
o	Others
a	All
r	Read permission
w	Write permission
x	eXecutable permission

chmod command

- Symbolic access modes {u,g,o} / {r,w,x}
  - example: `chmod +r` *`file`*
- Octal access modes

octal	read	write	execute
0	No	No	No
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes
4	Yes	No	No
5	Yes	No	Yes
6	Yes	Yes	No
7	Yes	Yes	Yes

Changing Ownership, Group

- To change the owner of a file:
  - `chown <owner> <file(s)>`
  - `chown <owner:group> <file(s)>`
  - `-R` recursive option available
- To change the group of a file
  - `chgrp <group> <file(s)>`
  - `-R` recursive option available

Unix File Structure/Permissions

From your home directory

```
> ls -l
```

`public_html` may be in different color than most entries

```
> ls public_html
```

Note: no / at end

```
> ls -l public_html

> ls -l /home/courses/cs297/
```

Assignment for Wednesday

- Practice UNIX commands
  - `script` command
- Exploring UNIX commands