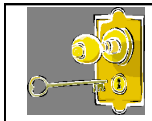
Review: Unix Commands

- How do we find out more information about Unix commands?
- Name some Unix commands and what they do
- How can I execute a command that I executed earlier?

Today

- Security
- File System commands

UNIX SECURITY



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…

How are Users & Groups used?

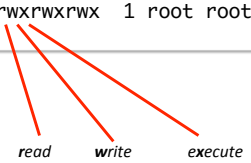
- Used to determine if file or process operations can be performed:
  - Can a given file be read? written to?
  - Can this program be run?
  - Can I use this piece of hardware?
  - Can I stop a particular process that's running?

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

A simple example

```
$ ls -l /bin
lrwxrwxrwx  1 root root  7 Feb 3  2016 /bin -> usr/bin
$
```



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@fred:cs397$ ls -lrth
total 12K
drwxr-x---. 17 sprenkle cs397 4.0K Jan 10 15:22 turnin
drwxr-s---.  3 sprenkle cs397 4.0K Jan 11 12:02 handouts
drwxr-xrwt.  2 sprenkle cs397 4.0K Jan 11 12:13 shared
```

permissions owner group size date modified file name

Unix Permissions

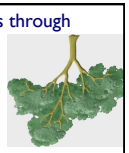
- Categories: owner, group, others
- Permissions: read, write, execute

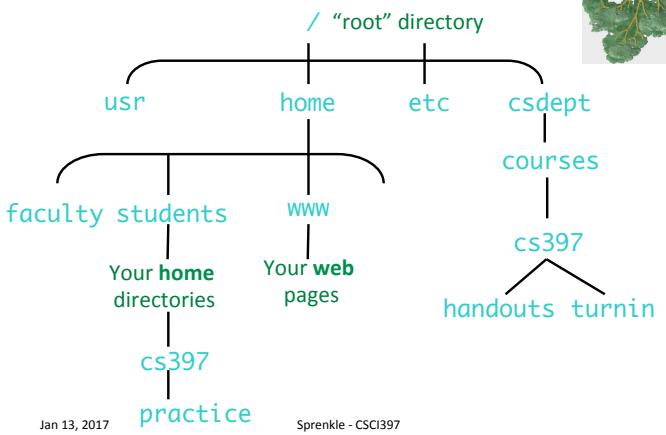
```
sprenkle@fred:cs397$ ls -lrth
total 12K
drwxr-x---. 17 sprenkle cs397 4.0K Jan 10 15:22 turnin
drwxr-s---.  3 sprenkle cs397 4.0K Jan 11 12:02 handouts
drwxr-xrwt.  2 sprenkle cs397 4.0K Jan 11 12:13 shared
```

permissions owner group size date modified file name

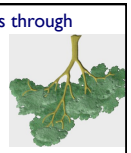
- What are the permissions on the file `tmp`?
- In the permissions, how can we distinguish between an executable file and directory?
- What does it mean for a file to be executable?

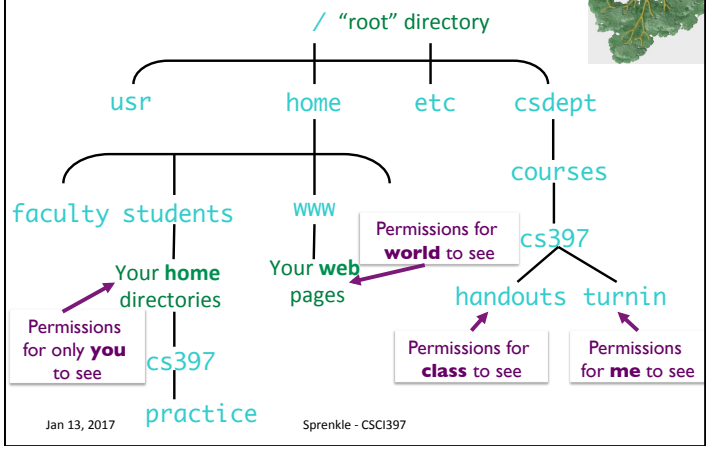
(Partial) Linux File Structure





(Partial) Linux File Structure





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 /csdept/courses/cs397/
```

Let's change the permissions on the handouts directory

MORE FILE COMMANDS

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

Try Out These Examples

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

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@fred ~]$ du -s ~/public_html/
5450740 /home/faculty/sprenkle/public_html/

[sprenkle@fred ~]$ du -sh ~/public_html/
5.2G    /home/faculty/sprenkle/public_html/
```

Try out on your public_html directory

Managing Disk Space

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

```
sprenkle@fred:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        5.9G     0  5.9G   0% /dev
tmpfs           5.9G  168K  5.9G   1% /dev/shm
tmpfs           5.9G  1.7M  5.9G   1% /run
tmpfs           5.9G     0  5.9G   0% /sys/fs/cgroup
/dev/sda3       102G   15G   82G  16% /
tmpfs           5.9G   48K  5.9G   1% /tmp
/dev/sda1       477M  164M  284M  37% /boot
hydros:/home    493G  128G  340G  28% /home
hydros:/local    99G  8.3G   86G   9% /csdept
tmpfs           1.2G   16K  1.2G   1% /run/user/42
tmpfs           1.2G   32K  1.2G   1% /run/user/0
tmpfs           1.2G  8.0K  1.2G   1% /run/user/1501
tmpfs           1.2G     0  1.2G   0% /run/user/1656
```

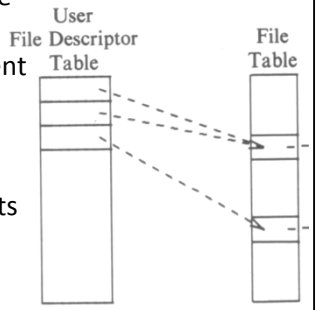
Timing Commands

- Often, you want to record when something happened or how long something takes
- date
  - Prints out system's current time
  - Lots of formatting options
    - Example: date +'%A, %B %d, %Y'
- time <simple command>
  - Measures command's resource use

FILE SYSTEM INTERNALS

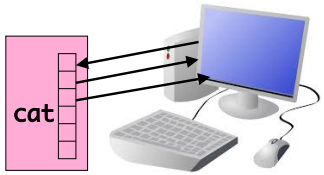
The File Descriptor Table

- Each process contains a table of files it has opened
- Inherits open files from parent
- Each open file is associated with a **number** or **handle**, called a **file descriptor** (fd)
- Each entry of this table points to an entry in the *open file table*
- Always starts at 0



Standard in/out/err

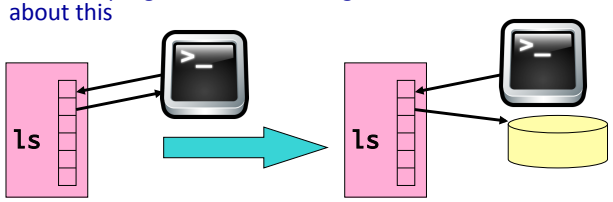
- The first three entries in the *file descriptor table* are special by convention:



- Entry 0 is for *input*
- Entry 1 is for *output*
- Entry 2 is for *error* messages

Redirection

- Before a command is executed, the input and output can be changed from the default (terminal) to a file
  - Shell modifies file descriptors in child process
  - The child program knows nothing about this



Redirection of input/ouput

- Redirection of output: >
  - Example:`$ ls > my_files`
  - Can save output from one of your programs
- Redirection of input: <
  - Example: `$ wc < input.data`
- Append output: >>
  - Example: `$ date >> logfile`
- Bourne Shell derivatives: `fd`>
  - Example: `$ ls 2> error_log`

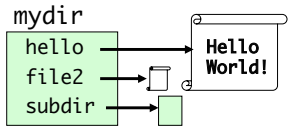
Redirecting Output

- Save output from a program
  - `> java OlympicScore > score.out`
  - Redirected `stdout` to `score.out`
  - `stderr` would still go to terminal
- To redirect `stderr` to file as well
  - `> java OlympicScore >& score.out`

LINKS

Links

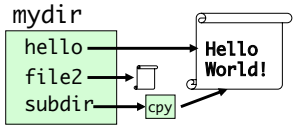
- Directories are lists of files and directories
- Each directory entry *links* to a file on the disk



- Hard links: Two different directory entries can link to the same file
  - Essentially gives same file another name
  - In same directory or across different directories
  - Cannot make a hard link to a directory

Links

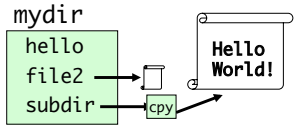
- Directories are lists of files and directories
- Each directory entry *links* to a file on the disk



- Two different directory entries can link to the same file
  - In same directory or across different directories
- Moving a file does not actually move any data around
  - Creates link in new location
  - Deletes link in old location
- `ln` command: `ln <target> <dest>`

Links

- Directories are lists of files and directories
- Each directory entry *links* to a file on the disk

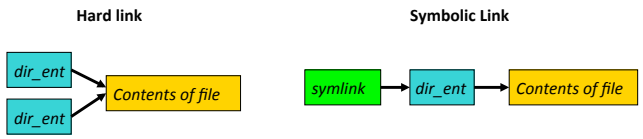


- Two different directory entries can link to the same file
  - In same directory or across different directories
- Moving a file does not actually move any data around
  - Creates link in new location
  - Deletes link in old location
- `ln` command: `ln <target> <dest>`

Jan 13, 2017 Sprenkle - CSCI397

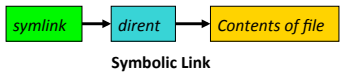
Symbolic links

- **Symbolic** links are different than regular links (often called **hard links**)
  - Created with `ln -s`
- Can be thought of as a directory entry that points to the ***name*** of another file

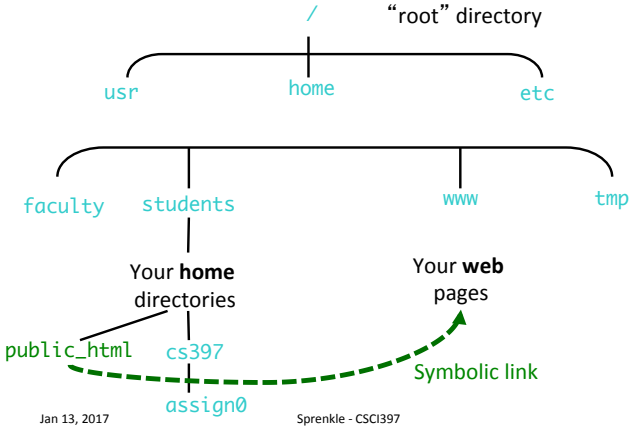


Symbolic links

- **Symbolic** links are different than regular links (often called **hard links**)
  - Created with `ln -s`
- Can be thought of as a directory entry that points to the ***name*** of another file
- Does not change link count for file
  - When original deleted, symbolic link remains
- They exist because
  - Hard links don't work across file systems
  - Hard links only work for regular files, not directories



(Partial) Linux File System



Practice

- Create a symbolic link to your `turnin` directory in your home directory