CS197U: A Hands on Introduction to Unix

Lecture 3: UNIX Operating System Organization

Tian Guo
CICS, Umass Amherst

Reminders

- Assignment 2 is due THURSDAY 09/24 at 3:45 pm
- Directions are on the website

- If you have questions about the assignment
  - I'll be in the Edlab Thursday from 1:00 to 3:00 pm
  - Email me

- If you plan to withdraw from the class, come talk to me.

Quick Quiz

- `head -n X [file]` --- prints out the top X lines of “file”
- `grep “string” [file]` --- print only the lines in “file” that contain “string”

- What is the difference between these commands?
  - `head -n 100 file.txt | grep “UMass”`
  - `grep “UMass” file.txt | head -n 100`

- Are these the same? Would one run faster than the other?
  - `sort lots_of_data.csv | grep “1973”`
  - `grep “1973” lots_of_data.csv | sort`

Quick Quiz

Answers:

- `grep “print” *.cpp | head -n 5`
- `sort phoneNumbers.txt >> contacts.txt`

- Write the commands to:
  - Display the first five lines that contain the word `print` in any .cpp file in the current directory
  - Sort the contents of the file phoneNumbers.txt and append it to the file contacts.txt
- Describe a situation when you would use the `head` utility instead of `cat`
- Describe a situation when you would use the `less` utility instead of `cat`
- Describe a situation when you would use the `cat` utility instead of `less`

Working with Paths

- `elnux3> pwd`

`/home/username/1/2/3/4/5/6/7/8`

- What does this do? What does it **not** copy?

`elnux3> cp ../../* ../../../../`

Doesn't copy any sub directories in 6

- What's the difference to this?

`elnux3> cp -r ../../* ../../../../`

Copies everything in 6 and below (7 and 8)

- What is the difference between these commands?

`elnux3> cd`

`elnux3> cd .`

`elnux3> cd ~`

Use . and ~ with other commands like cp

Useful tips

- `head -n 3 days.txt | sort | uniq| grep –c “Friday”`

- If the output of a lot of piped commands doesn’t make sense to you…
- … decompose:
  - `head -n 3 days.txt | less`
  - `head -n 3 days.txt | sort | less`
  - `head -n 3 days.txt | sort | uniq| less`
  - Look at intermediate output after each pipe
  - also useful for “debugging”

Search for a command in history

- Use UP and DOWN keys in terminal
  - Takes too long to find a command used a long time ago
- `history | less`
  - Use `/<search_string>`, type `n` to go to next
  - copy command
    - Good, but still slow
- `history | grep “<search_string>”`
  - Use `!<number>` to execute. Note: `<number>` is the number listed to the left of the command.
- Use reverse incremental search (in `bash` shell only)
  - `Ctrl + r`, start typing command
  - Type `Ctrl + r` again to find next match
  - Fast!

Password-less SSH login

- Connect to remote hosts without entering your password every time
  - e.g., connect to `elnux` machines in Edlab
- From a Unix shell only (Linux or Mac OS X)
  - Create SSH keys
    - run `ssh-keygen -t rsa`
    - copy `~/.ssh/id_rsa.pub` to `~/.ssh/authorized_keys` on remote host
      - `scp ~/.ssh/id_rsa.pub username@elnux1.cs.umass.edu:~/.ssh/authorized_keys`
    - ~ is short for your home directory
- From Windows
  - Different procedure for putty
  - http://www.tecmint.com/ssh-passwordless-login-with-putty/

The plan...

- Last time:
  - Overview of useful Linux commands
    - reading files: `cat`, `head`, `tail`, `less`, `grep`
    - editing files: `vim`, `sort`, `uniq`
    - manipulating files: `mv`, `cp`, `rm`

- This time we will:
  - **Learn about OS structure**
  - Explore the Linux file system
  - Learn about file permissions

In this class:

- Discuss basics of OS structure and function
  - If you're interested in more details, consider taking the Operating Systems class!
- Discuss role of operating system and responsibilities of the Linux kernel

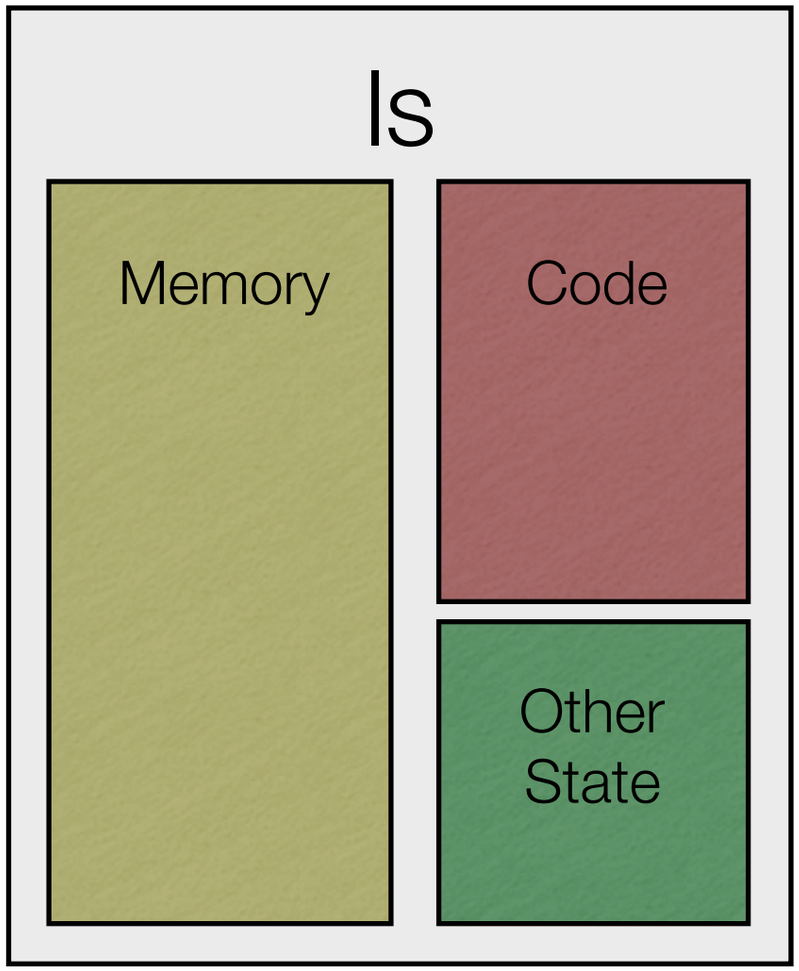
Things the OS does

- Layer between user applications and hardware
- Goal: Hide complexity of hardware from applications/users

- Responsible for:
  - Taking care of multitasking (time-sharing) in an intelligent way
  - Managing resources (memory, network, CPU, devices, etc)
  - Separation (security, stability)

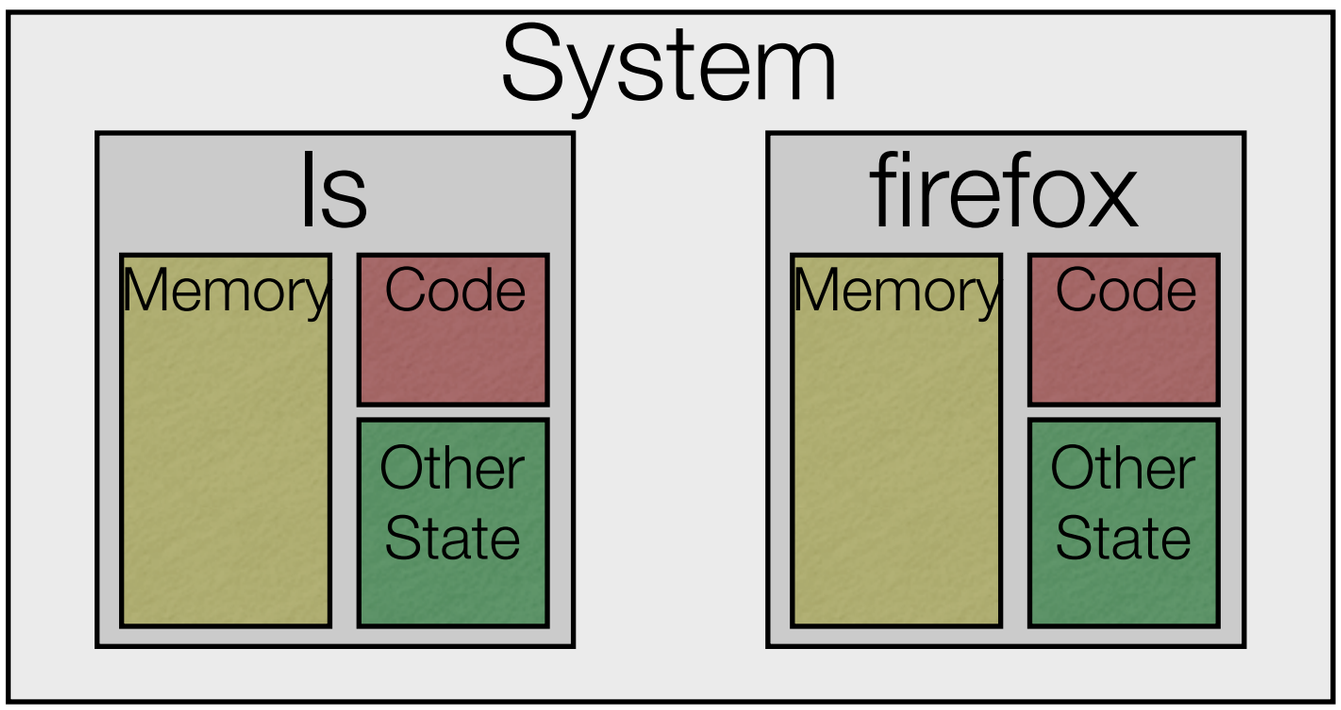
Basic concept: Processes

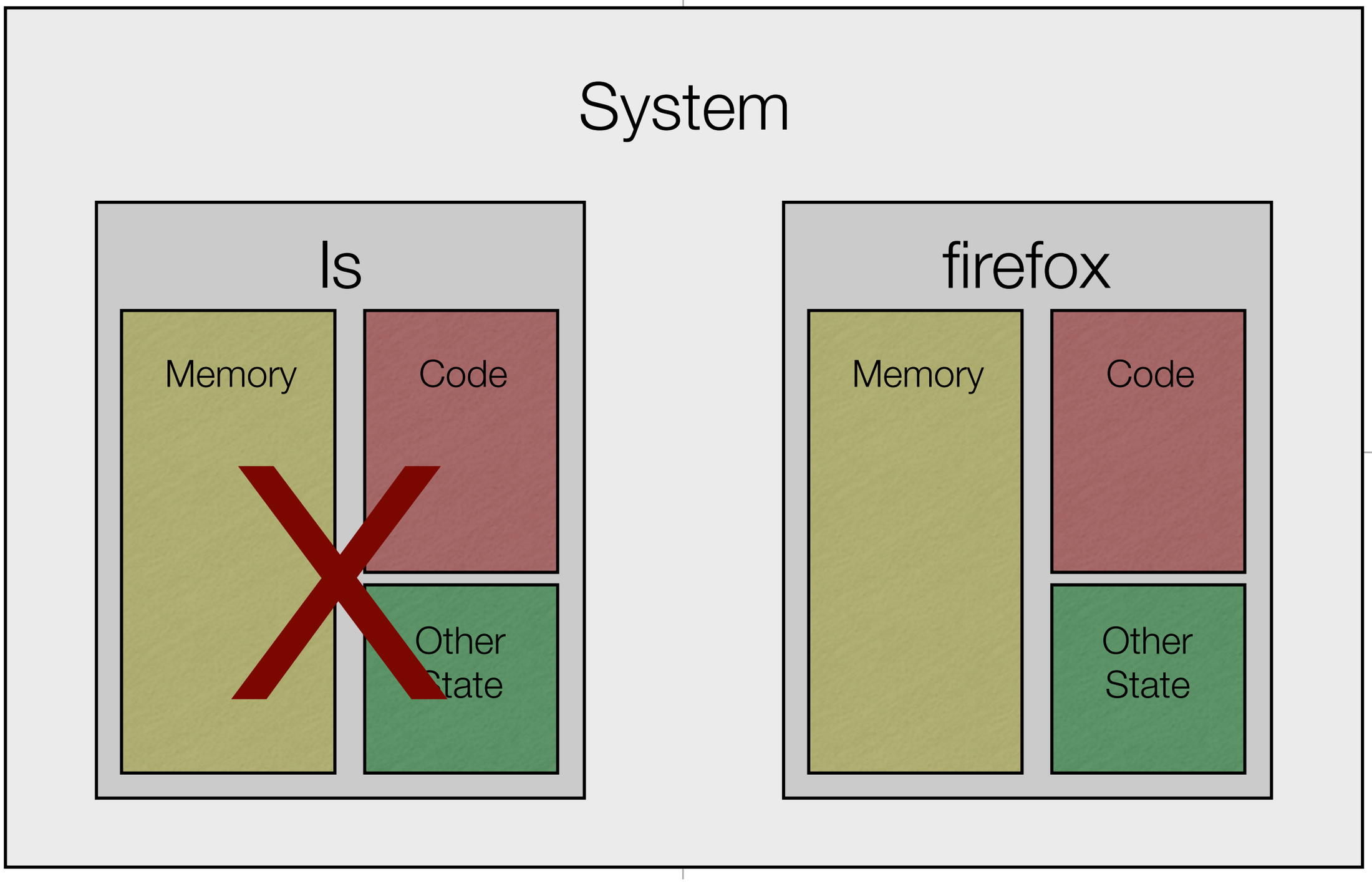
- Instance of a program, which is being executed
- Isolated from other processes
- All processes appear to execute simultaneously
  - OS makes it seem like each process is running alone



Basic concept: Processes

- Processes don’t need to reserve time from other processes; each one executes as if it had its own CPU
  Isolated from other processes
- A process’s memory is isolated and protected from other processes.
- A process accesses the disk, network, etc with system calls





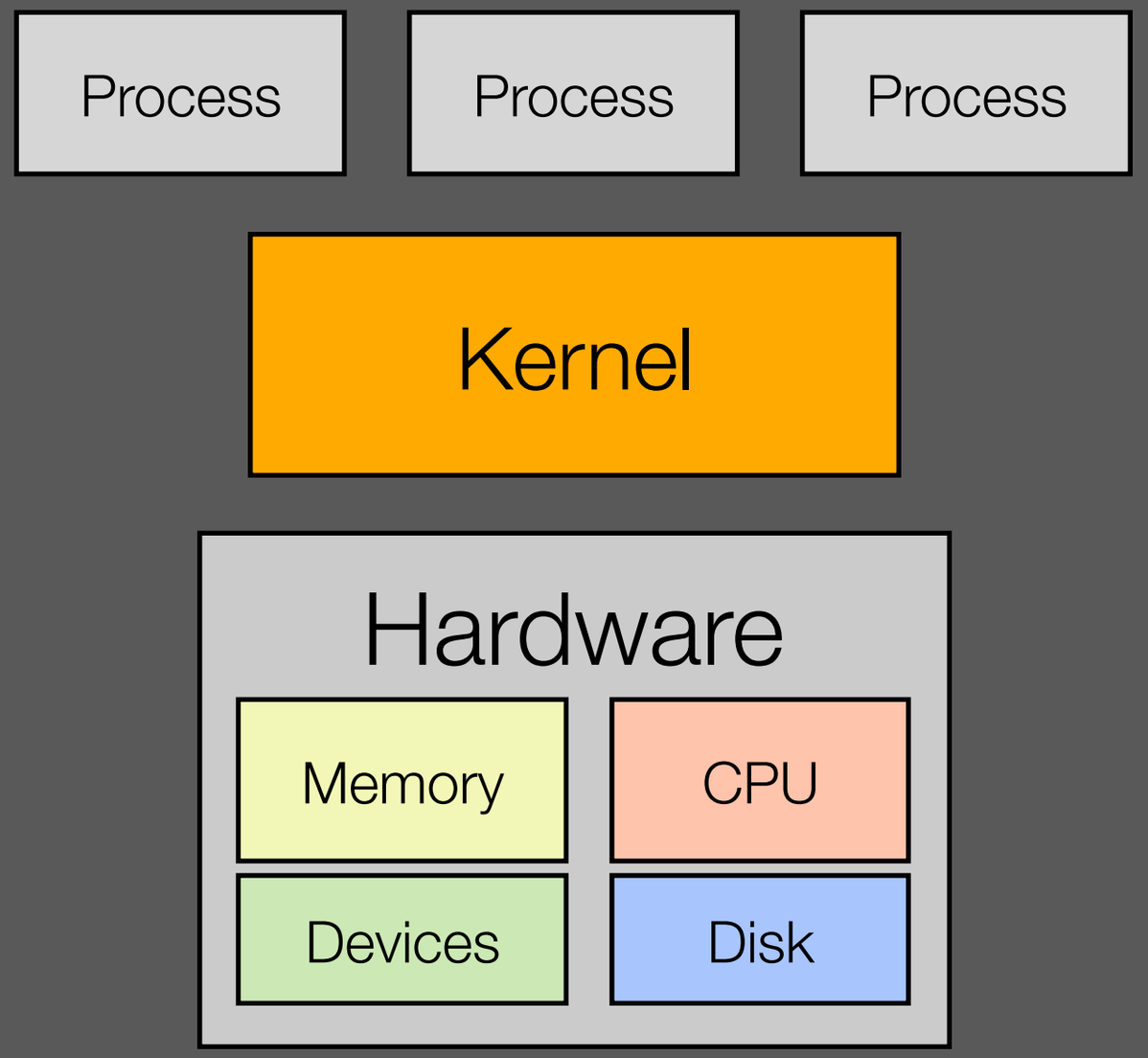
If one application crashes or misbehaves, this should not affect other processes

How the system works

- The OS must make each process appear to have its own CPU, address space, etc
- The OS uses abstract APIs to access storage, network, devices, etc
  - Hardware can change, OS still works
  - Any number of processes can exist
- But hardware only provides fixed resources (one CPU, one memory space, one disk, and low-level devices)

The Kernel

- The Kernel is a master program that manages resources
- The Kernel executes directly on the hardware
- The Kernel then runs all other processes



Main kernel tasks

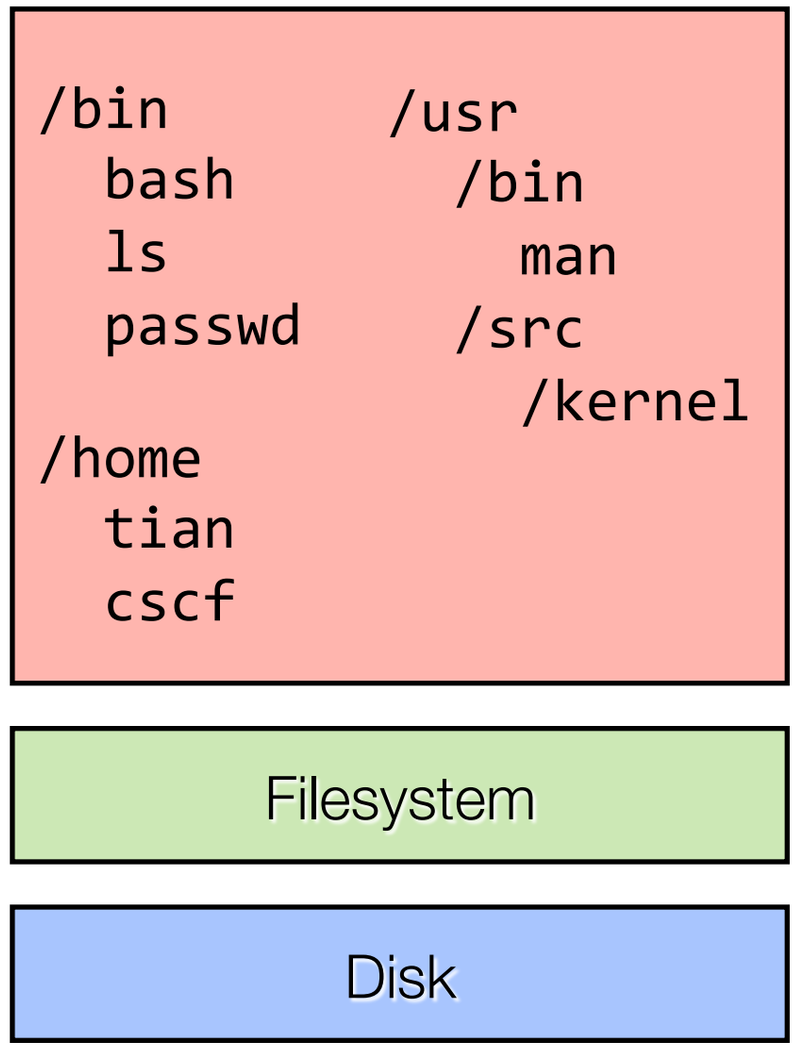
- Implement multitasking
- Implement memory management and protection
- Device interfacing
- Translate high-level system calls into low-level device commands

The plan...

- This time we will:
  - Learn about OS structure
  - **Explore the Linux file system**
  - Learn about file permissions

Filesystem

- Abstraction of actual storage device
  - Data isn't stored as files, but as blocks
  - Filesystem makes organization and use easier
- Implements a named, hierarchical storage system
- Divides up a single volume into many directories
  - Each stores different types of information



Filesystem basics

- A regular file is a string of bytes
  - File extension (e.g. “.txt” or “.doc”) is only meaningful to the user
- A directory contains other files or directories
- Special files, such as device files also exist

Filesystem directories

- Directories allow the filesystem to be hierarchical

- Directories can contain many other directories, but all directories have only one parent

- The directory / is the root directory

Paths

- Paths name a file
  - Either relative to the current directory
    - `cs197u/<username>/file.txt`
  - From an absolute location
    - `/courses/cs100/cs197u/cs197u/assignments/assign-2`

- Paths starting with `/` are absolute paths, from the root directory:

- Paths starting with `~/` are relative to the user's home directory
  - Can refer to another user's home directory with:
    `~username/`

- All other paths are relative to the current directory

Filesystem Tour - /

- /bin/ - executable files
- /sbin/ - Privileged Executables
- /dev/ - Devices
- /etc/ - System configuration
- /home/ - User home directories
- /lib/ - libraries
- /opt/ - third-party software
- /sys/ - Kernel, device drivers, etc
- /usr/ - other OS programs and data
- /var/ - Logs, databases, rapidly-changing files

Filesystem Tour - `/etc/` - configuration files

- /etc/rc, /etc/rc.D/ - System startup program; started by kernel during boot process
- /etc/passwd, /etc/passwd.db - user info, minus passwords (usually)
- /etc/shadow.passwd - user info, with encrypted passwords
- /etc/groups - group info
- a lot of .conf files

Filesystem Tour - `/opt/`, `/usr/` - user apps

- /.../bin/ - executable files
- /.../sbin/ - Executable files which execute at higher privileges
- /.../lib/ - libraries
- /.../libexec/ - Additional private configuration
- /.../src/ - source code
- /.../man/ - man pages
- /.../local/ - local additions to standard distributions
- /.../share/ - Additional information (examples, etc.)

Filesystem Tour - `/home/` - user directories

- Directories named after individual users
- sometimes has bin/, lib/, ... subdirectories
- User-specific configuration data files often start with .
- .profile - executed when starting a shell
- .ssh/ - ssh config data
- other . files for specific programs

Filesystem Tour - `/dev/` - devices

- Devices appear as files in UNIX
    - Here you'll find hard drives, CD drives, etc.
- Don't need to be physical devices, some have special functions
    - /dev/null - Null device, ignores all reads, discards all writes
    - /dev/zero - produces zeros when read
    - /dev/random, /dev/urandom - produces random numbers when read
- Other devices do specific things (hard drives can be directly read)

Looking for things in the file system

- `find` – prints the path to a file
  - `find <dir> -name ‘*.txt’ -print`
  - Use for finding a specific file in some directory
  - Very useful, many flags
- locate – prints the path to a file; very good for finding **system files**

```
> locate passwd
/etc/passwd
/etc/alternatives/vncpasswd
/etc/pam.d/chpasswd
/etc/pam.d/passwd
/etc/security/opasswd
/usr/NX/scripts/restricted/nxpasswd.sh
/usr/bin/gpasswd
/usr/bin/grub-mkpasswd-pbkdf2
/usr/bin/kpasswd
```

The plan...

- This time we will:
  - Learn about OS structure
  - Explore the Linux file system
  - **Learn about file permissions**

File ownership

Use `ls -l` to see file ownership details

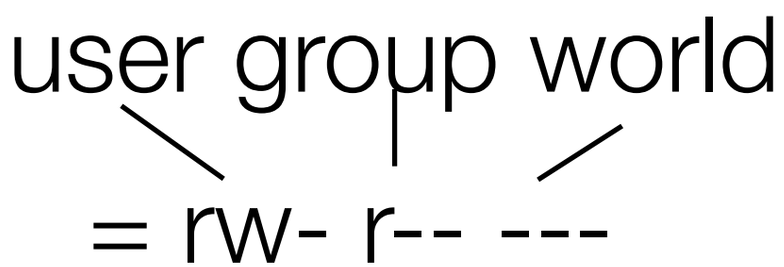
- Each file is owned by a **group** and a **user**

- A **user** is an account for an individual person
  - "root" is a special account for the system administrator
  - Each file has one **user** that owns it
- A **group** is a collection of users
  - A group can hold multiple users
  - Each user can be in multiple groups
  - Type `groups` to see what groups you are a part of
  - Used to let a user share files with others
  - Each file is associated with one **group**

File permissions

- Can control access for the user, group, or “world” (everyone else)
- Possible actions:
  - **Read** file contents
  - **Write** to a file (edit or overwrite)
  - **Execute** a program or open a directory

	read	write	exec
user	Y	Y	N
group	Y	N	N
world	N	N	N



Directory permissions

- Directories have same type of permission
  - Execute is permission to open the directory
  - Only applies to directory itself -- individual files will have own settings

	read	write	exec
user	Y	Y	Y
group	Y	N	Y
world	Y	N	Y

= rwx r-x r-x

	read	write	exec
user	Y	Y	Y
group	Y	Y	Y
world	Y	Y	Y

= rwx rwx rwx

Setting Permissions

- Command to change: `chmod <setting> <filename>`

<setting>: <x><y><z>

<x>: u - user
g - group
o - world
a - all

<y>: + add
- remove

<z>: r - read
w - write
x - execute

Examples:

`chmod u+r file.txt`	Add user read permission for file.txt
`chmod g-x file.txt`	Remove group execute permission for file.txt
`chmod o-w file.txt`	Remove world write permission for file.txt

Setting Permissions

- Command to change: `chmod <setting> <filename>`

<setting>: <x><y><z>

<x>: u - user
g - group
o - world
a - all

<y>: + add
- remove

<z>: r - read
w - write
x - execute

Exercises:

	Add user & group write permission for file.txt
	Remove world read,write & execute permission for file.txt
	Add world,group & user read,write and execute permission for file.txt

Setting Permissions

- Command to change: `chmod <setting> <filename>`

<setting>: <x><y><z>

<x>: u - user
g - group
o - world
a - all

<y>: + add
- remove

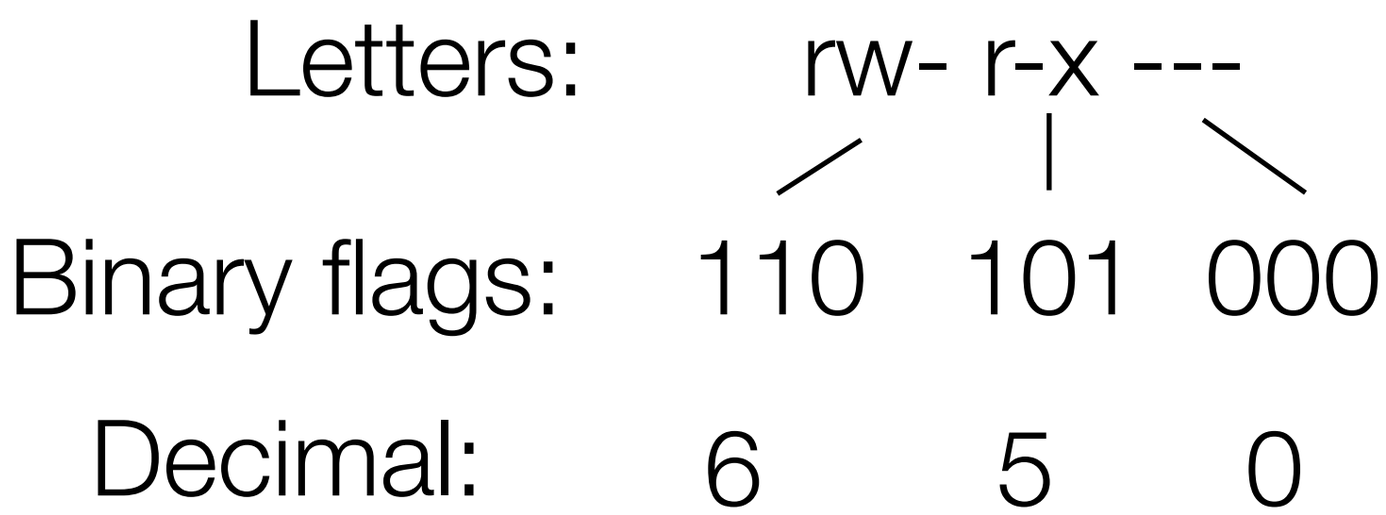
<z>: r - read
w - write
x - execute

Exercises:

`chmod gu+w file.txt`	Add user & group write permission for file.txt
`chmod o-rwx file.txt`	Remove world read,write & execute permission for file.txt
`chmod ogu+rwx file.txt` `chmod a+rwx file.txt`	Add world,group & user read,write and execute permission for file.txt

Setting Permissions

- Permissions can also be represented as numbers



Permissions code: 650

Command to change permissions: `chmod 650 filename`

Letters:	r--	r--	rwx
Binary flags:	100	100	111
Decimal:	4	4	7

37

Letters:	rwx	rwx	rwx
Binary flags:	111	111	111
Decimal:	7	7	7

Summary

- An OS is responsible for
  - Allocating resources to processes
  - Providing protection
  - Coordinate access to I/O devices

Remember Assignment 2 is Due THURSDAY before class!

- The Linux filesystem has a lot of stuff in it
  - As a user you'll generally only have to deal with your home directory

- Linux provides a set of file permissions to limit access
  - See Chapter 5 in book for more advanced options
    - Changing what user owns a file, what group owns a file, changing default permissions, etc.

Lecture 3 review

Command	Description
`history`	Print a list of previously run commands
`chmod`	Change permissions for a directory or file
`find`	Print a path to a file
`locate`	Print a path to a file