

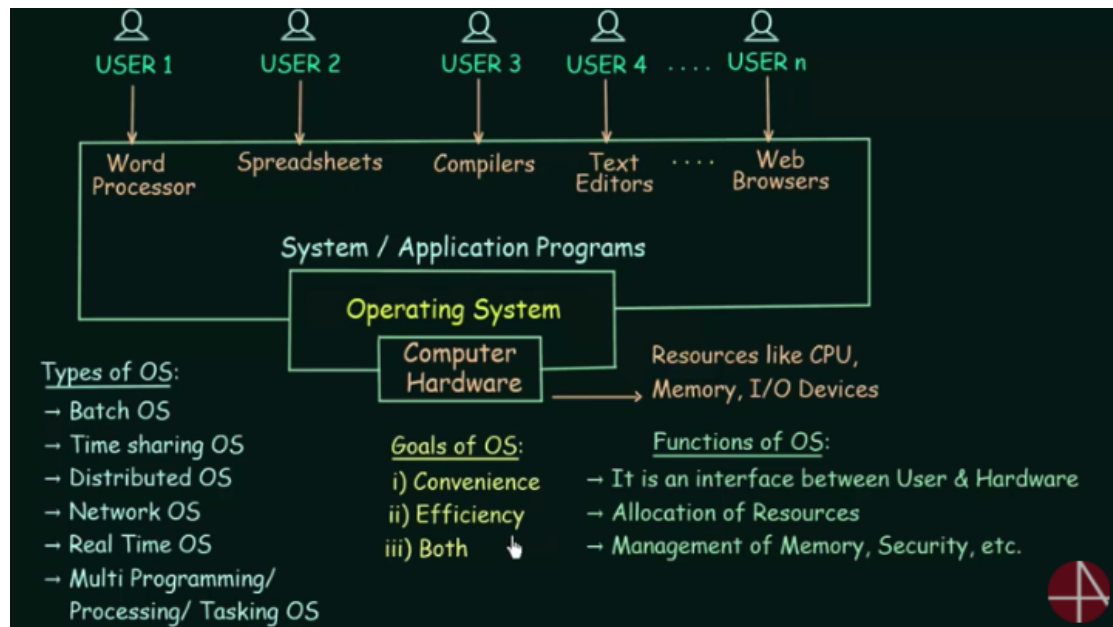
CSC369 Week 1 Notes

Hyungmo Gu

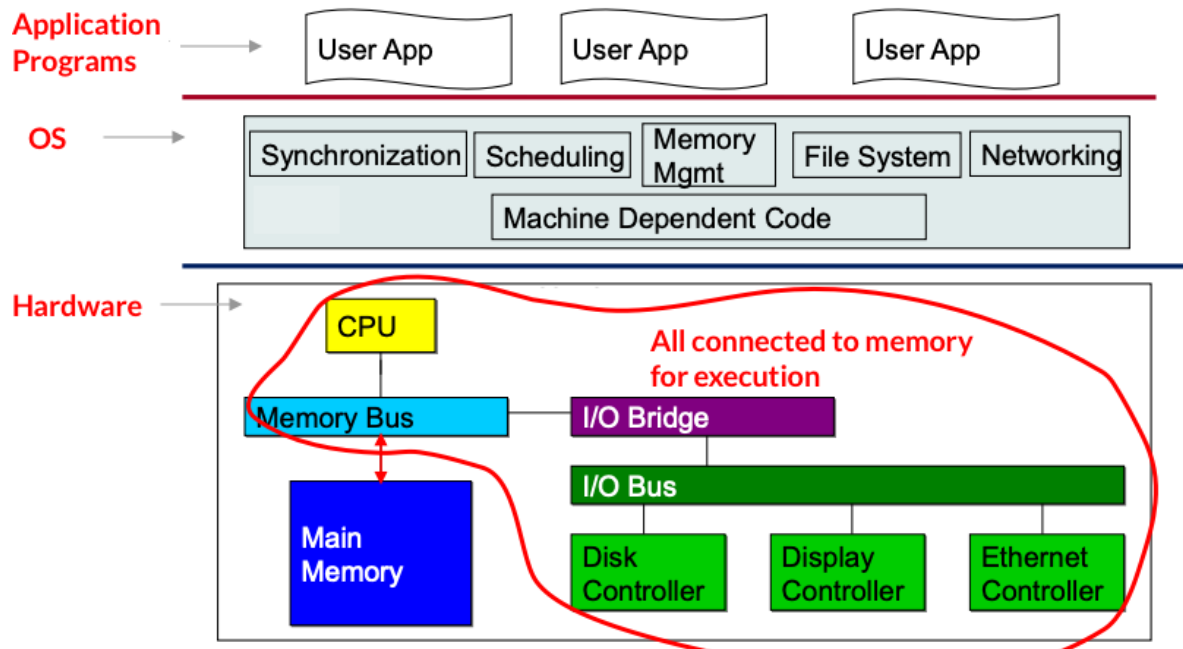
May 19, 2020

1 Intro to OS

- What is Operating System
 - is the program that manages the computer hardware
 - is the software layer between user applications and hardware
 - is used for
 - * Allocation of resources
 - * Management of memory, security, etc.

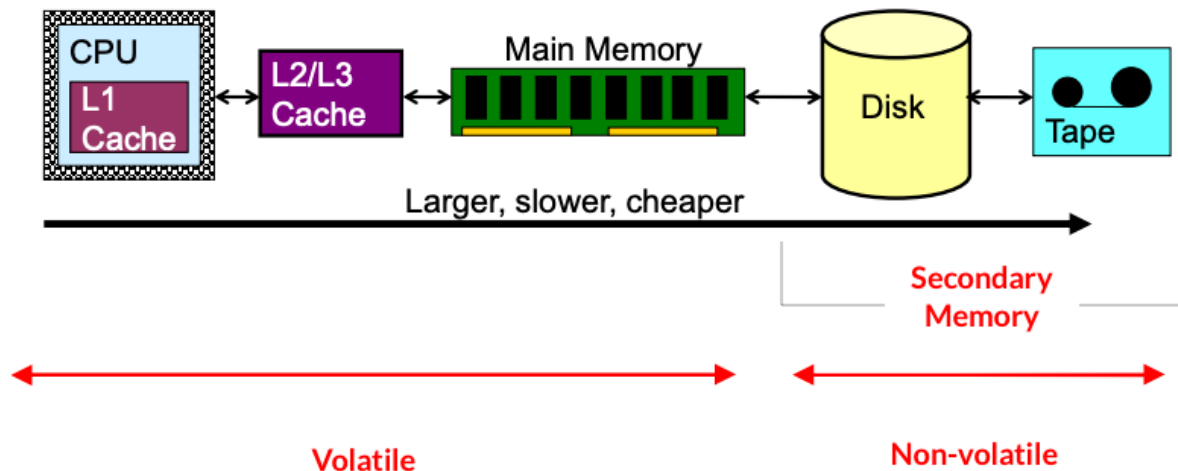


- Overview of Computer System



- All hardware devices are connected through common **bus** and are loaded to memory for execution.
- **Synchronization:** to ensure orderly acces to the shared memory

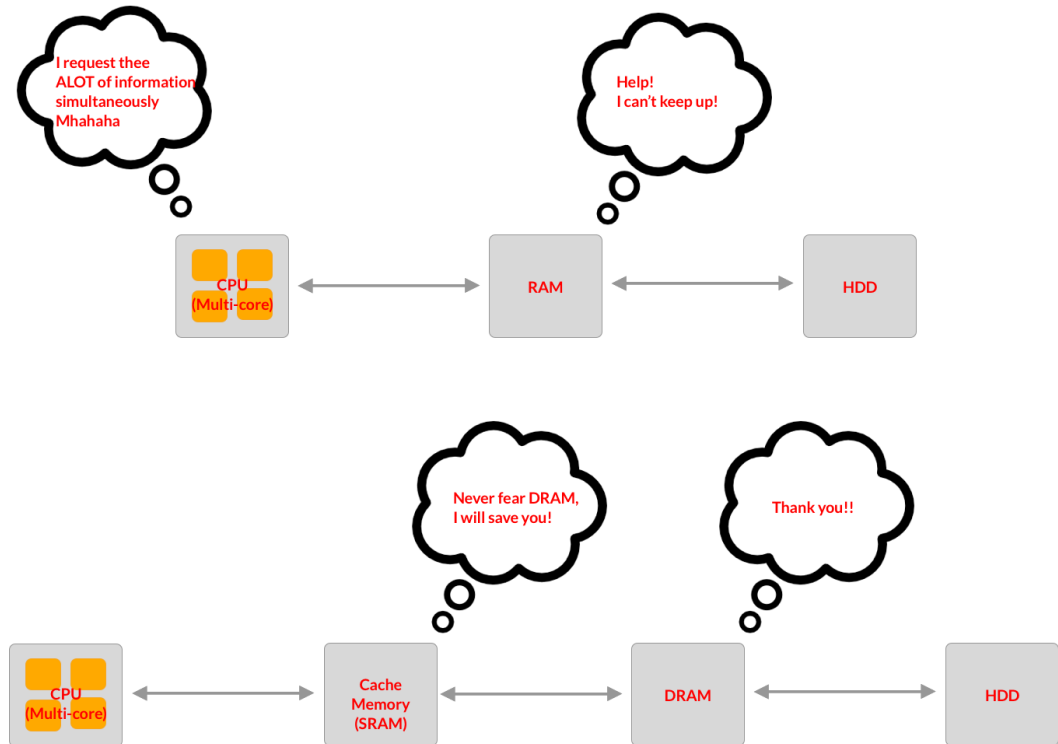
- Storage Hierarchy / Storage Structure



- **Volatile** → Loses contents when power is removed
- **Non-volatile** → Retains contents even when power is removed

- Caching / Cache Memory

- Is also called **Static Random Access Memory (SRAM)**
- Is more costly
- Hides performance differences when large access-time gap exists between two levels
 - * Quad-quare requesting RAM for information



- More can be found here

- Concurrency

- Is execution of several instruction sequences at same time
 - * i.e, CPU and device controllers
- **Interrupt:** are signals sent to the CPU by external devices, (usually I/O devices)
 - * It's like telling 'Hey CPU, please stop this process, and do y instead, since this is more important'
 - * i.e. Network Packet has arrived, Disk I/O complete occurred
- **System Call:** are interrupt signals sent by software
 - * Is a programmatic way of a program requesting for service to kernel of operating system
 - * i.e. Accessing a hard-disk drive
- **IMPORTANT:** An operating system is an event-driven program.

2 Process Threads

- Part 1: The Process Concept
 - **Process:** is a program in execution
 - **Threads:** is the unit of execution within a process.

$$\text{Thread} = \frac{\text{Job}}{\text{Unit of Work}} \quad (1)$$

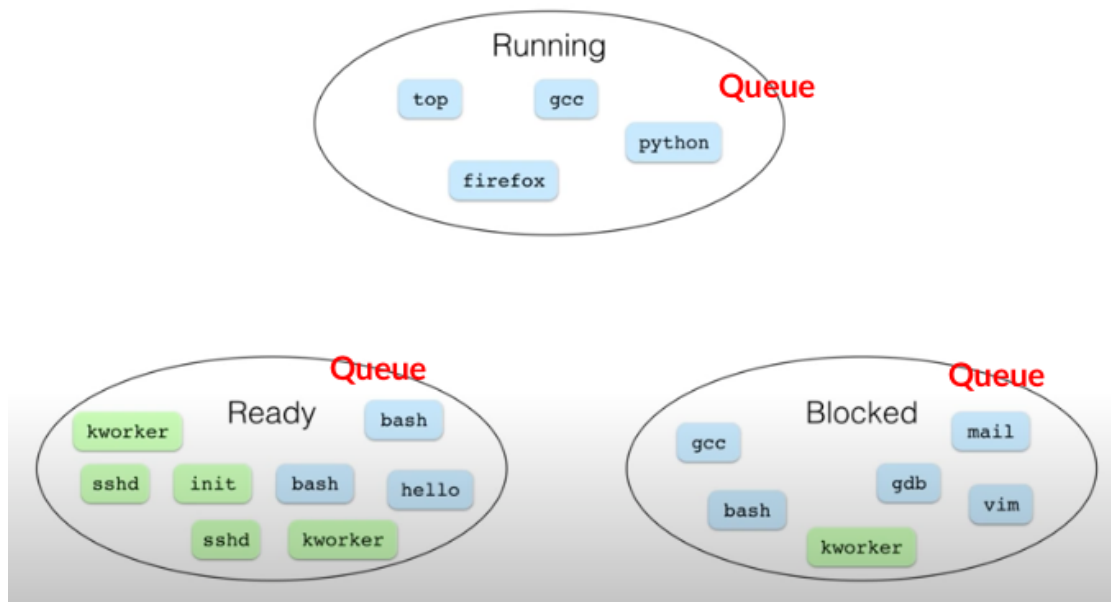
* A process can have anywhere from one thread to many threads

- Process Data Structure (PCB)
 - Is called Process Control Block
 - Is OS data structure representing each process
 - Generally includes
 1. Process State
 - * (Ready, running, blocked)
 2. Program Counter
 - * Is an address that indicates the line of code that has to be executed next
 - * i.e. the next line of code i need to execute is line 2 :)

```
1 print("Hello World");
2 print("Hi World!") //<- Line 2
```
 3. CPU Register ***Need to come back*
 4. CPU Scheduling Information
 - * Priority of process
 - * Higher the priority → executed first
 5. Memory Management ***Need to come back*
 6. I/O Status Information
 - * Is list of input output devices assigned to this process
 - * Is used during execution
 - * i.e. Sound, Mouse, Keyboard
- State Queues
 - Is a part of **process scheduling**
 - * keeps CPU busy at all times to deliver minimum response time for all programs

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	% GPU	GPU Time	PID	User
WindowServer	43.7	40:07.03	11	66	2.4	20:38.71	330	_windowserver
Code Helper (Renderer)	37.3	8:23.49	32	24	0.0	0.00	2584	moegu
Code Helper (GPU)	18.3	4:02.04	8	41	0.3	2:37.24	2581	moegu
Adobe CEF Helper	11.5	21:36.14	10	112	0.6	5:41.56	884	moegu
kernel_task	9.6	17:02.14	185	1210	0.0	0.00	0	root
com.docker.hyperkit	8.1	15:16.38	18	285	0.0	0.00	1389	moegu
Activity Monitor	6.8	9.60	5	2	0.0	0.00	7295	moegu
Code	4.5	2:06.91	45	1	0.0	0.00	2580	moegu
Adobe CEF Helper	3.0	5:43.96	15	181	0.0	0.00	1232	moegu
Google Chrome	2.1	23:40.61	40	18	0.0	0.00	569	moegu
hidd	1.7	2:15.82	6	0	0.0	0.00	231	_hidd
launchd	1.5	55.70	6	0	0.0	0.00	1	root
sysmond	1.4	2.05	3	0	0.0	0.00	5728	root
mdworker_shared	1.1	0.49	4	0	0.0	0.00	7266	moegu
launchservicesd	0.8	7.18	7	1	0.0	0.00	210	root
Creative Cloud	0.8	1:38.47	24	65	0.0	0.00	814	moegu
Be Focused	0.7	1:32.46	7	13	0.0	0.00	2931	moegu
Google Chrome Helper (GPU)	0.5	22:35.98	11	3	0.7	10:51.38	753	moegu
mds	0.4	38.98	9	4	0.0	0.00	197	root
vpnkit-bridge	0.4	32.03	15	59	0.0	0.00	1354	moegu
tcdd	0.3	1.13	3	0	0.0	0.00	5694	root
mds_stores	0.3	2:46.31	7	1	0.0	0.00	391	root

- * Here, processes in queue are switched so frequently that user can interact with each program simultaneously while running
 - i.e, listening to music, typing and downloading a picture of a cute puppy all at the same time
- Has one state queue for each process state
 - * Job Queue, Ready Queue, Waiting Queue, Blocking Queue



- PCBs And State Queues
 - Process created → OS allocates PCB → Initializes it → Places it on Ready Queue
 - Process terminated → PCB deallocated
- Context Switch

- Switches the CPU to another process on interrupt, saving the state of the old process and loading the saved state of the new process until done
 - * i.e. Loading a music in Google Music Player
- Previous process resumes executing when done

The top screenshot shows the Activity Monitor window with the CPU tab selected. The process list is as follows:

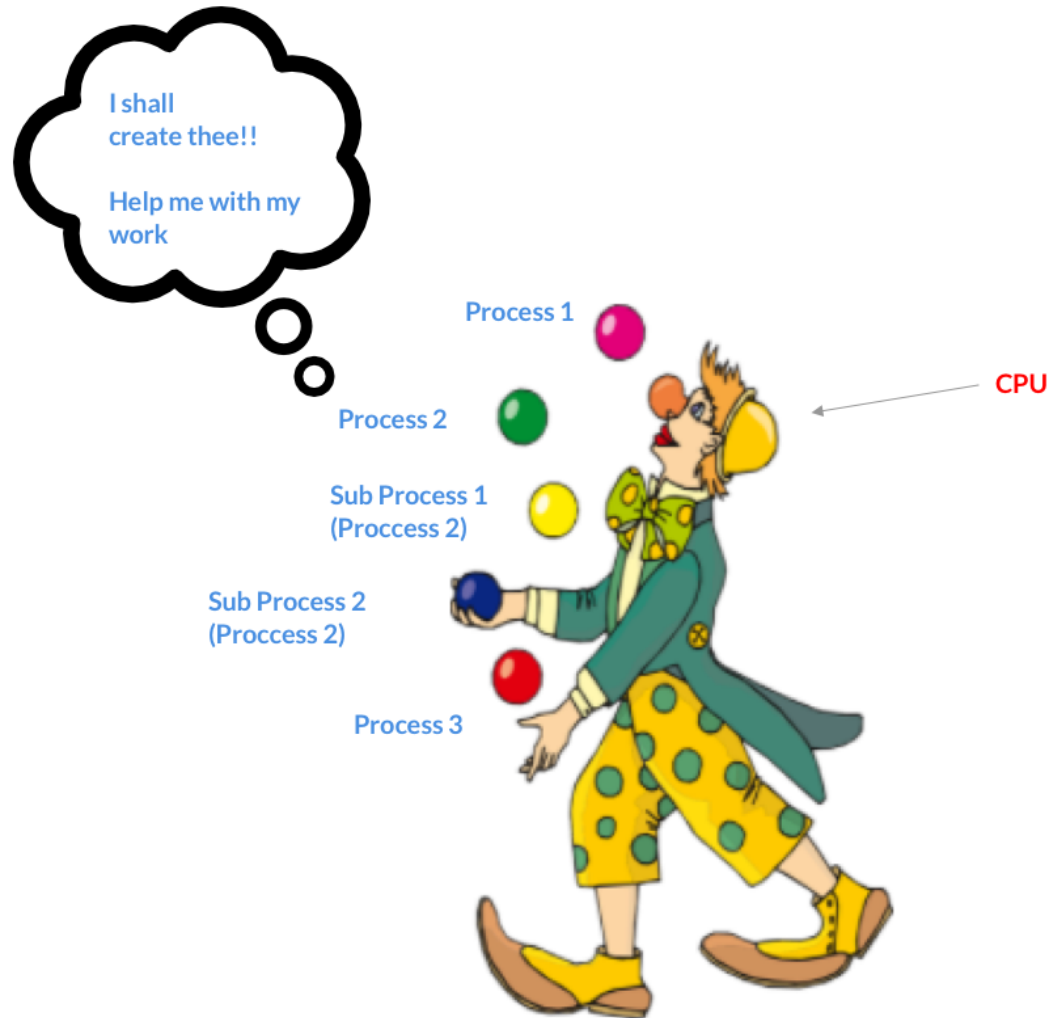
Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	% GPU	GPU Time	PID	User
WindowServer	43.7	40:07.03	11	66	2.4	20:38.71	330	_windowserver
Code Helper (Renderer)	37.3	8:23.49	32	24	0.0	0.00	2584	moegu
Code Helper (GPU)	18.3	4:02.04	8	41	0.3	2:37.24	2581	moegu
Adobe CEF Helper	11.5	21:36.14	10	112	0.6	5:41.56	884	moegu
kernel_task	9.6	17:02.14	185	1210	0.0	0.00	0	root
com.docker.hyperkit	8.1	15:16.38	18	285	0.0	0.00	1389	moegu
Activity Monitor	6.8	9.60	5	2	0.0	0.00	7295	moegu
Code	4.5	2:06.91	45	1	0.0	0.00	2580	moegu
Adobe CEF Helper	3.0	5:43.96	15	181	0.0	0.00	1232	moegu
Google Chrome	2.1	23:40.61	40	18	0.0	0.00	569	moegu
hidd	1.7	2:15.82	6	0	0.0	0.00	231	_hidd
launchd	1.5	55.70	6	0	0.0	0.00	1	root
sysmond	1.4	2.05	3	0	0.0	0.00	5728	root
mdworker_shared	1.1	0.49	4	0	0.0	0.00	7266	moegu
launchservicesd	0.8	7.18	7	1	0.0	0.00	210	root
Creative Cloud	0.8	1:38.47	24	65	0.0	0.00	814	moegu
Be Focused	0.7	1:32.46	7	13	0.0	0.00	2931	moegu
Google Chrome Helper (GPU)	0.5	22:35.98	11	3	0.7	10:51.38	753	moegu
mds	0.4	38.98	9	4	0.0	0.00	197	root
vpnet-bridge	0.4	32.03	15	59	0.0	0.00	1354	moegu
tcdd	0.3	1.13	3	0	0.0	0.00	5694	root
mds_stores	0.3	2:46.31	7	1	0.0	0.00	391	root

The bottom screenshot shows the same Activity Monitor window, but the 'Activity Monitor' process has moved up in the list. A red arrow points to its new position with the text 'Hey, this moved up!'.

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	% GPU	GPU Time	PID	User
WindowServer	31.2	49:31.01	10	38	1.9	24:38.93	330	_windowserver
Activity Monitor	10.5	1:07.77	5	2	0.0	0.00	7295	moegu
Adobe CEF Helper	10.4	25:59.20	10	75	1.0	6:49.66	884	moegu
Code Helper (GPU)	7.5	5:43.22	8	15	1.1	3:51.30	2581	moegu
com.docker.hyperkit	6.8	18:06.55	18	187	0.0	0.00	1389	moegu
Code Helper (Renderer)	4.1	1:11.28	19	19	0.0	0.00	7422	moegu
kernel_task	3.9	19:33.70	185	466	0.0	0.00	0	root
Google Chrome	3.1	25:50.85	34	4	0.0	0.00	569	moegu
Adobe CEF Helper	2.9	6:52.54	15	125	0.0	0.00	1232	moegu
sysmond	2.3	40.40	3	0	0.0	0.00	5728	root
Be Focused	1.9	1:54.22	7	9	0.0	0.00	2931	moegu
Code Helper (Renderer)	1.7	11:21.09	30	10	0.0	0.00	2584	moegu
hidd	1.5	2:42.94	5	0	0.0	0.00	231	_hidd
Code	1.4	2:57.23	47	2	0.0	0.00	2580	moegu
AppleUserHIDDrivers	1.0	1.34	3	0	0.0	0.00	8411	_driverkit
AppleUserHIDDrivers	0.9	5.09	3	0	0.0	0.00	8410	_driverkit
AppleUserHIDDrivers	0.7	0.97	2	0	0.0	0.00	8413	_driverkit
Creative Cloud	0.7	1:57.41	25	45	0.0	0.00	814	moegu
Code Helper (Renderer)	0.6	13.62	15	14	0.0	0.00	7421	moegu
launchservicesd	0.6	12.99	7	0	0.0	0.00	210	root
AppleUserHIDDrivers	0.4	2.10	2	0	0.0	0.00	8412	_driverkit
contextstored	0.3	12.33	5	0	0.0	0.00	283	root

• Operations on Processes

- Process can have multiple new processes during the course of execution, or alone
- Process execute concurrently and must be created and deleted dynamically
 - * Wow, CPU is a master juggler



- Has two types of operations
 - * Process Creation
 - * Process Termination
- Process Creation
 - Is an operation that creates new processes by another process
 - * Child process → New Process
 - * Parent Process → Creating Process
 - Is achieved through **fork()** system call
 - Occurs in
 1. System Initialization
 2. A running process
 3. A user request
 4. Initialization of a bath job
 - Two possibilities exist on creation:

1. Parent executes with children in parallel
 2. Parent waits until some or all of its children are terminated
- More can be found here
- `fork()`
 - Duplicating Address Processes
 - Divergence
 -