Operating Systems

Dr. Naser Al Madi

Welcome back!
I hope you had a fun and productive break.

Happy new year!
Happy lunar new year!
Happy black history month!

Outline

- Syllabus
- What is an Operating System and why do we need it?
- OS types
- What is a process? (process vs. Program)

Google interview prep guide

Topics covered in this course

Operating Systems: You should understand processes, threads, concurrency issues, locks, mutexes, semaphores, monitors and how they all work. Understand deadlock, livelock and how to avoid them. Know what resources a process needs and a thread needs. Understand how context switching works, how it's initiated by the operating system and underlying hardware. Know a little about scheduling and the fundamentals of "modern" concurrency constructs.

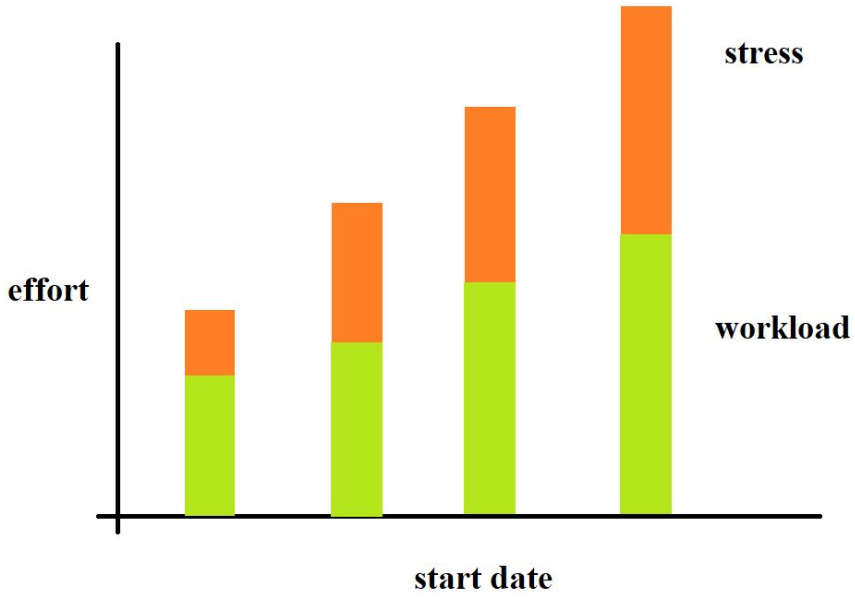
Syllabus

Projects

1. Scheduling 1: Visualization of **non-preemptive** scheduling algorithms in Python
2. Scheduling 2: Visualization of **preemptive** scheduling algorithms in Python
3. Scheduling 3: Visualization of **Advanced** scheduling algorithms in Python
4. Scheduling 4: Adding scheduling algorithm to **Xv6** in C (don't worry, we will learn it together)
5. Multithreading: frequency counter - reddit comments in Python
6. Synchronization: **Semaphores** and dining philosophers problem in Xv6 in C.
7. Memory management: Visualization of **Page and Frame Replacement** Algorithms in Python
8. Disk management: Visualization of **disk seek** Algorithms in Python
9. Mass storage: Benchmarking **RAID** configurations.

Common pitfalls

Starting project late



Not asking for help

- I am happy to answer your questions anytime.
- Send me an email if you have a quick question, also I am happy to meet in-person or on Zoom, if needed.

What is an operating system?

What is an operating system? (informally)

- Layer between hardware and software
- Manages resources like CPU, main memory, storage, devices, communication and more.
- It can be tailored to a specific task, like:
  - Running a webserver
  - Running games (PlayStation/Xbox)
  - Running software for a spacecraft
  - Running a general purpose software on personal computers/mobile devices

The Linux Story

- Linus Torvalds created Linux when he was 21
- Could not afford Unix, so he wrote Linux
- Linux is a recursive name: **L**inux **I**s **N**ot **U**ni**X**
- Shared the code as an Open-source project, and the community went crazy.
- To manage the large number of programmers that contribute to Linux, he wrote Git (Yup, the same tool used by GitHub).
- Linux is the most popular “operating system” in the world (used in Android).

A small collection of operating systems based on Linux:

Source: Link

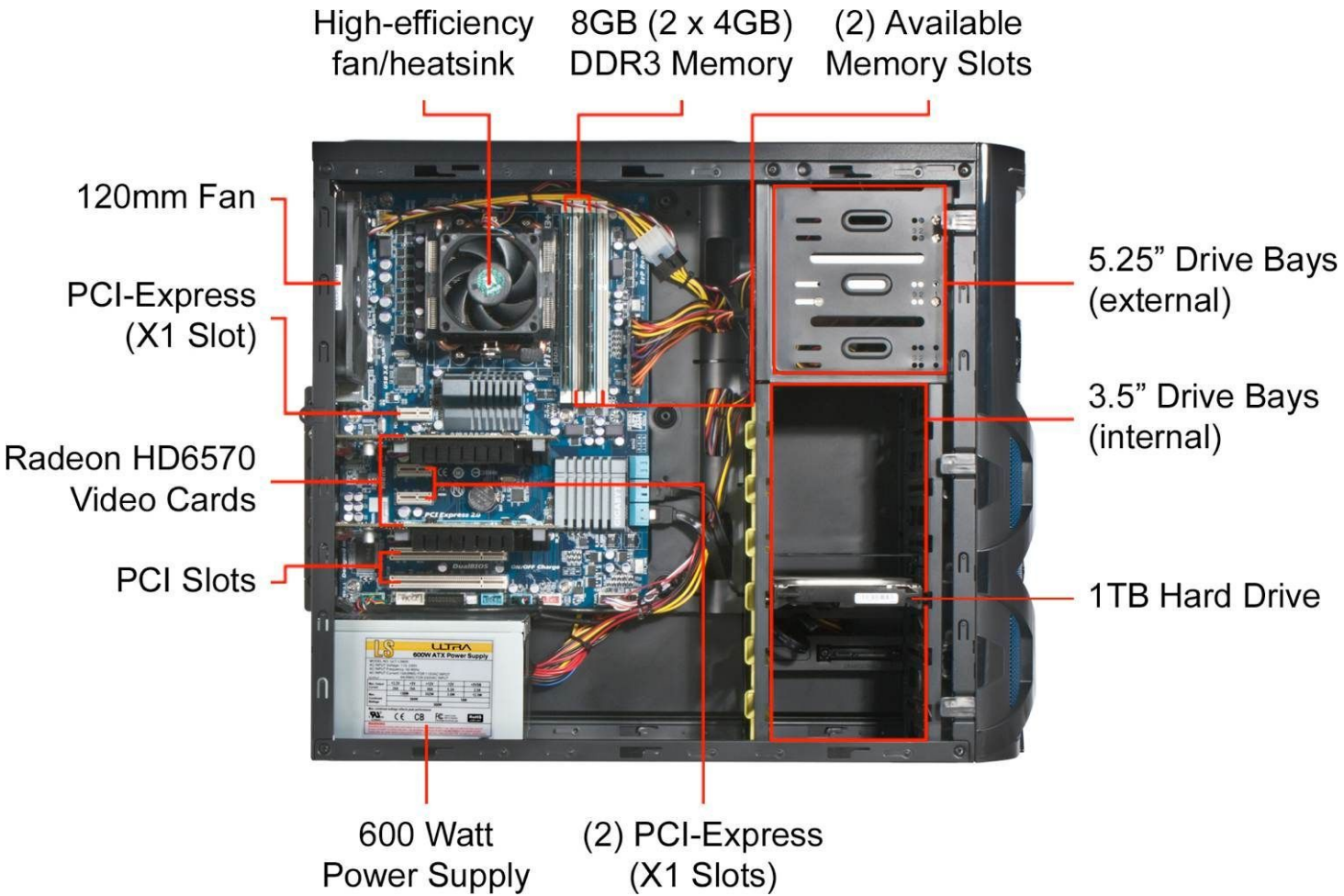
Operating System Definition

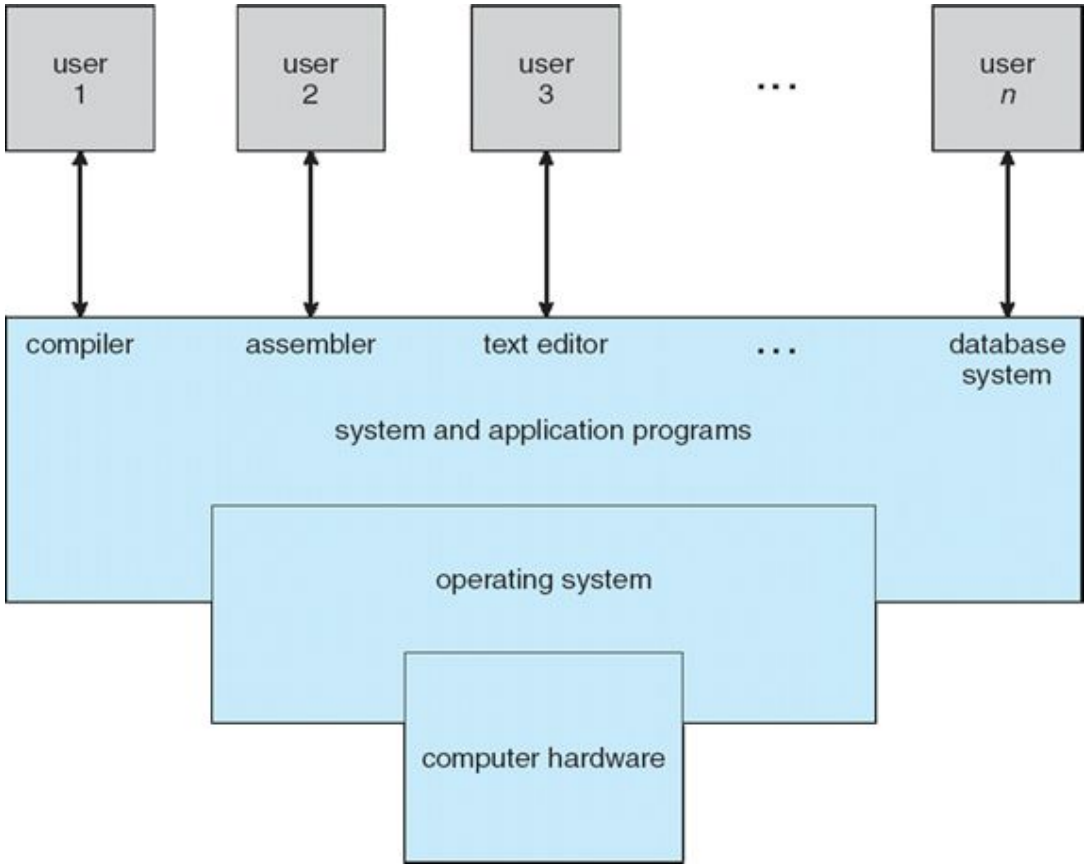
- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
- A program that acts as an **intermediary** between a user of a computer and the computer hardware
- “The one program running at all times on the computer” is the **kernel** (heart and soul of OS)

Computer System Structure

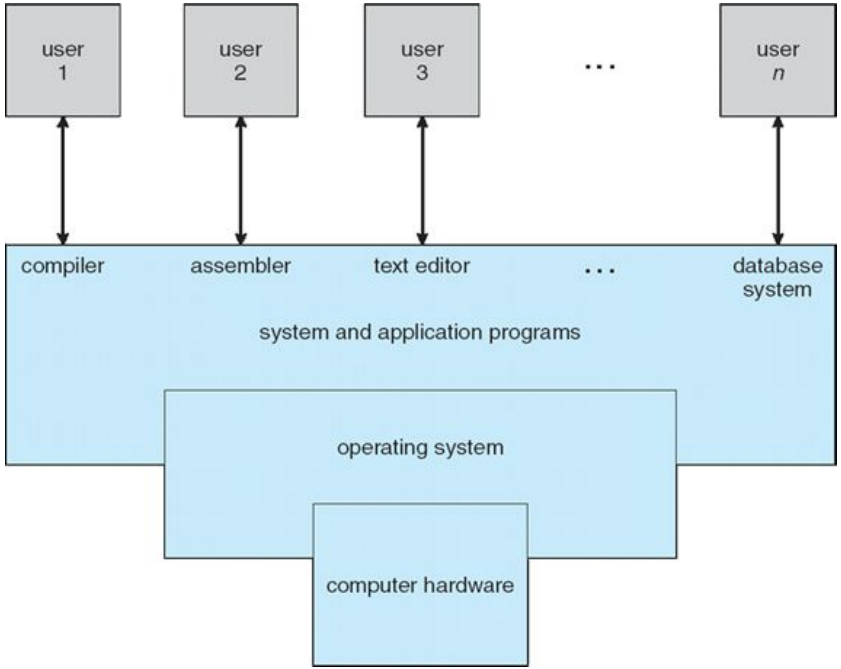
Computer system can be divided into four components:

- Hardware – provides basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- Users
  - People, machines, other computers


High-efficiency fan/heatsink
8GB (2 x 4GB) DDR3 Memory
(2) Available Memory Slots
120mm Fan
PCI-Express (X1 Slot)
Radeon HD6570 Video Cards
PCI Slots
5.25” Drive Bays (external)
3.5” Drive Bays (internal)
1TB Hard Drive
600 Watt Power Supply
(2) PCI-Express (X1 Slots)


user 1
user 2
user 3
...
user n
compiler
assembler
text editor
...
database system
system and application programs
operating system
computer hardware

Why do we need it?



Can we run a computer without an OS?

- You could run a computer without an operating system (bare metal like Arduino)
- Your program must have knowledge of specific hardware components
- The program can not be too complicated (can’t easily use keyboard/mouse/monitor…etc)
- You can run only one program at a time (no concurrency)

OS advantages:

- Concurrency: allows multiple processes to run at the same time (seamlessly)
- Hardware utilization: makes best use of available hardware resources
  - CPU scheduling: time-sharing CPU/s to make progress on all running processes
  - Memory management: selecting how to organize memory and what to store at a given time
  - Disk management: allow for reliable long-term storage
  - Device management: allows for devices to be added/removed and used easily
- Platform: provide tools to run user programs without knowledge of specific hardware

OS types and their design philosophy

- Time Sharing operating systems
- Distributed operating systems
- Network operating systems
- Real-time operating systems

NON-Time Sharing operating systems

- MS DOS is not a time sharing OS
- It allows only one process to run at a time
- When the process is finished, you go back to the Command-line interface where you can run another:

```
Starting MS-DOS...

C:\>_
```

*Not a time sharing OS

NON-Time Sharing operating systems

- No concurrency
- CPU and hardware utilization is limited (CPU is fast, user is slow)
- Can’t provide a responsive interface (like GUI)

*Not a time sharing OS

Time Sharing operating systems

- Time Shared Operating System is also known as the Multi Tasking Operating System.
- Time-sharing operating systems implement CPU scheduling to deliver to every process a small piece of run time.
- CPU “juggles” processes really quickly, giving the illusion that all processes are running at the same (concurrency).

Android Windows 10 Microsoft Windows Ubuntu Linux macOS Chrome OS

What operating systems keep things running in space?

Source: article

What operating systems keep things running in space?

Real-time operating systems

Source: article

Real-time operating systems

Critical systems need:

- Perfect timing
- Reliability
- Handling multiple tasks concurrently
- Logging
- Fail safe

Source: article

SpaceX aborts Starlink 5 launch at T-0

Real-time operating systems

- an operating system (OS) for real-time applications that processes data and events that have critically defined time constraints.
- All processing must occur within the defined constraints (< 1ms)
- Event-driven systems switch between tasks based on their **priorities**, while time-sharing systems switch the task based on **time**.

Real-time operating systems

- an operating system (OS) for real-time applications that processes data and events that have critically defined time constraints.
- All processing must occur within the defined constraints (< 1ms)
- Event-driven systems switch between tasks based on their **priorities**, while time-sharing systems switch the task based on **time**.

Pathfinder (we will talk about it)

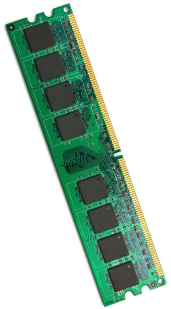
What is a process?
How does it really work?

Process Vs. Program

- A process is a program in execution. It is a unit of work within the system.
- Program is a passive entity, process is an active entity.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Process executes instructions sequentially, one at a time, until completion
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs

The life cycle of a process

CPU

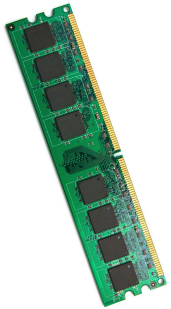


Main Memory
(RAM)

Disk

The life cycle of a process

CPU



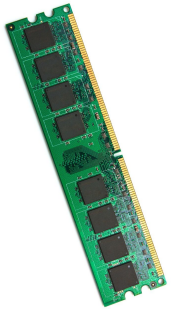
Main Memory
(RAM)

Disk

clickety click

The life cycle of a process

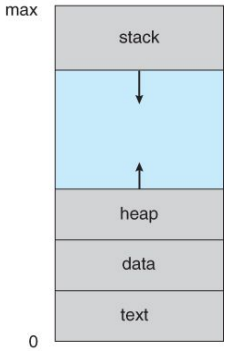
CPU



Main Memory
(RAM)

Disk

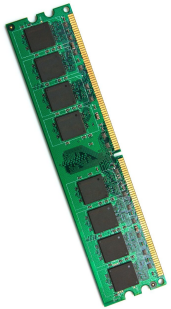
The OS loads the program instructions, data, heap, and stack into memory.



clickety click

The life cycle of a process

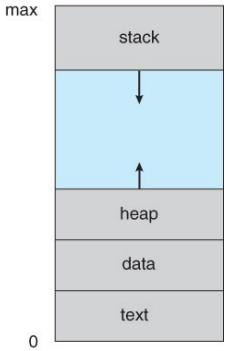
CPU



Main Memory
(RAM)

Disk

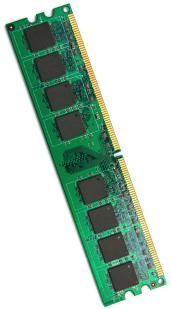
The OS loads the program instructions, data, heap, and stack into memory. (partial loading is possible)



clickety click

The life cycle of a process

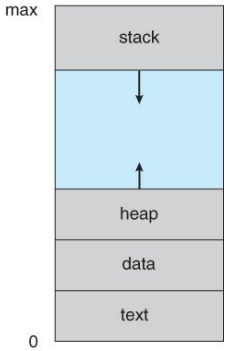
CPU



Main Memory
(RAM)

Disk

The OS also creates an internal view of this process, called Process Control Block or PCB



clickety click

Process control block

Process state: running, waiting, ready, finished, zombie

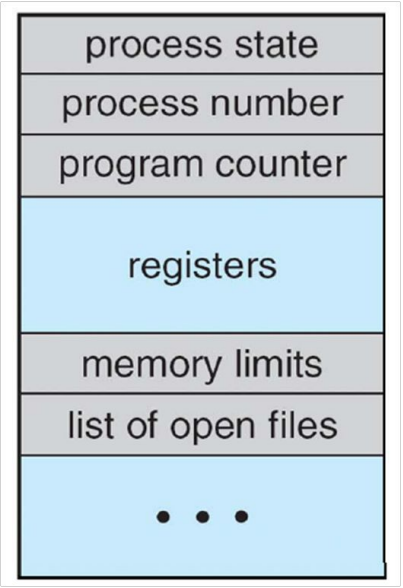
Process number: unique ID for each process

Program counter: a pointer to the next instruction

Registers: holds what was in the CPU before the process got interrupted last

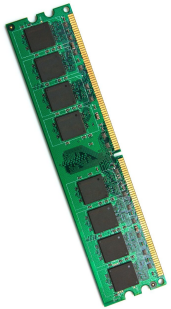
Memory limits: the start and end of the valid memory block for this process

Open files: lost of files open by this process



The life cycle of a process

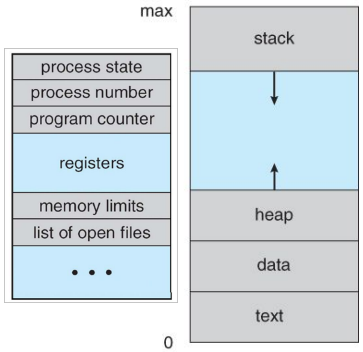
CPU



Main Memory
(RAM)

Disk

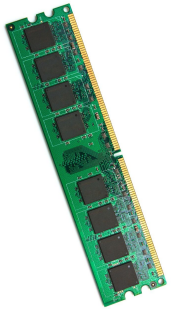
When the scheduler picks this process to run in the CPU, this process is loaded into CPU registers and the execution continues from the pointer to the next instruction.



clickety click

The life cycle of a process

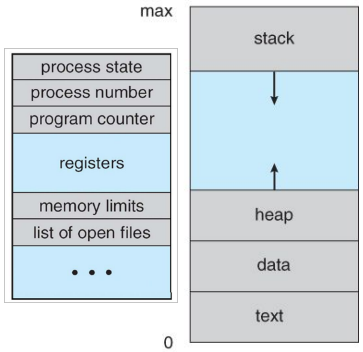
CPU



Main Memory
(RAM)

Disk

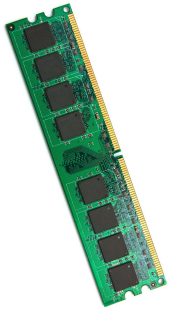
The process runs in the CPU for a little while…



clickety click

The life cycle of a process

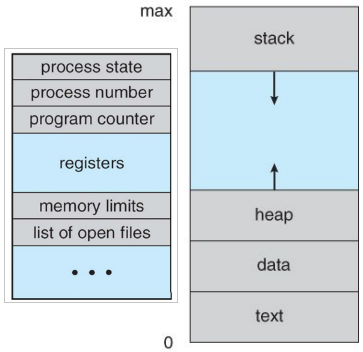
CPU



Main Memory
(RAM)

Disk

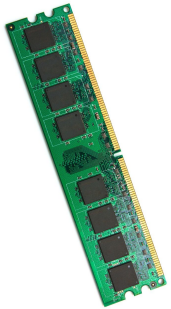
Then the scheduler picks another process to run, so this one is paused, the PCB is updated, and another process enters the CPU.



clickety click

The life cycle of a process

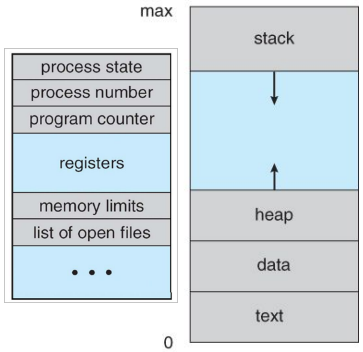
CPU



Main Memory
(RAM)

Disk

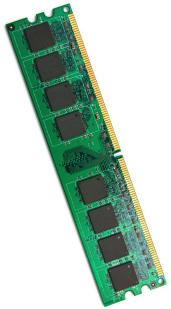
This repeats many times very quickly, giving the user the impression that multiple processes are running at the same time.



clickety click

The life cycle of a process

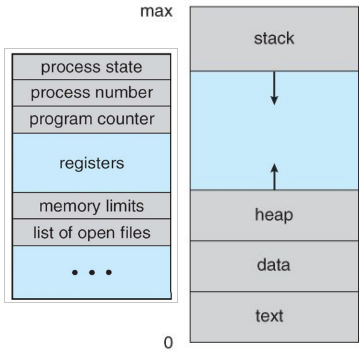
CPU



Main Memory
(RAM)

Disk

When the process is finished, the OS is responsible for releasing any memory occupied by this process, closing its files, and clearing the PCB.



clickety click

That was a simplified view

We didn’t take in consideration how to load open files, how moving the mouse interrupts this process, or how to read characters from keyboard into process and so on.

We will learn all that in details this semester.

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines:
  - What is in memory and when, optimizing CPU utilization and computer response to users
  - Which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Isn’t an OS a process? Who runs it?

- The **Kernel** is the OS process running at all times.

- **bootstrap** program is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

To be continued...

References:

Operating System Concepts, 9th Edition – Silberschatz, Galvin, and Gagne

Understanding Operating Systems, 8th edition - Ann McHoes and Ida M. Flynn