Date: 17/06/25

Introduction to Computer Systems

showvik.biswas@braco.ac.bd

4M128 (consultation).

Slack (next week).

Three easy pieces (favourite book).

Course Outline

Overview of OS
Process management
CPU scheduling
Threads
Process Synchronization
} mid

Memory management
} final

Before OS

The Punched Card Era

Data is encoded by punching holes at specific location on these cards.

Each card corresponds to one line of code.

Huge stacks required for a single program

Running multiple programs is totally out of the Question.

Same goes for I/O operations → output.

input

Solution

Program

↳ manage my programs

↳ allow multiple programs at a time.

↳ manage input/output operations on my behalf.

→ Something which works as a middleman between the hardware and my programs.

Job Automation

first we need batch processing.

To run program one after another is batch processing.

Able to switch from one job to another upon completion/failure.

A burden reduced!

Multiprogramming

keep multiple programs in memory and share the CPU between them.

Multiplexing

keep CPU busy always.

Time Sharing

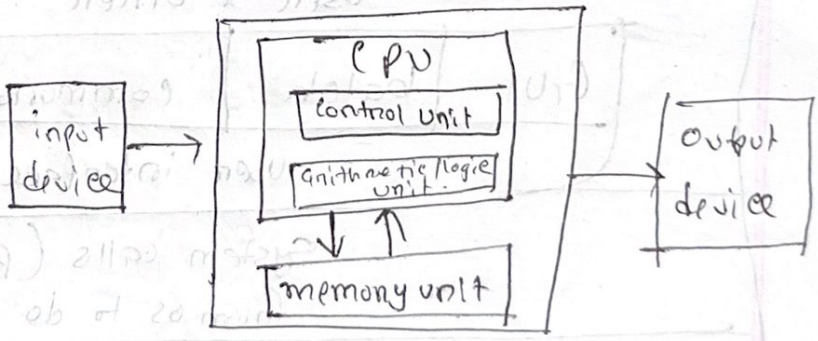
A logical extension of multiprogramming.

CPU running time is shared between different users, giving each the illusion of being the only person to run the computer.

Input / Output

most computers are organized according to the von neumann model.

for, interaction with its environment, I/O is essential.



Device drivers - key part of an OS.

Our OS needs to handle I/O operations and devices as well.

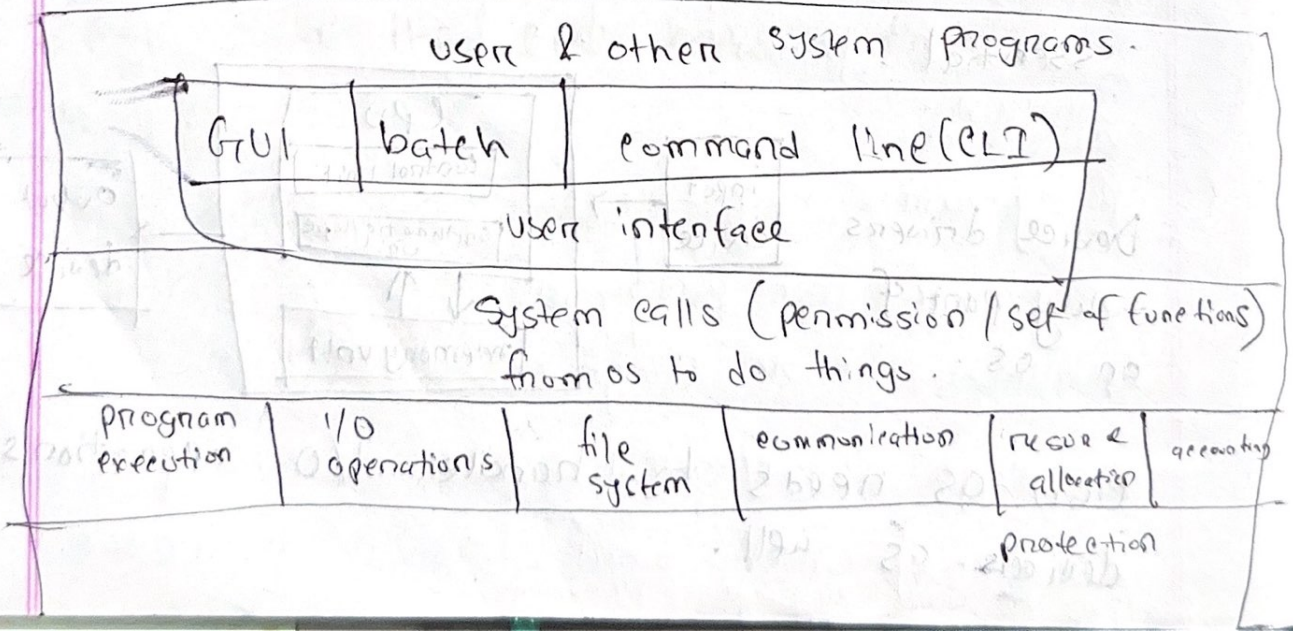
Problem with I/O

Compared to CPU operation, I/O operations are extremely slow.

A process heavily reliant on I/O operations might waste resources and disallow another CPU intensive process from running.

OS needs to able to smart enough to switch programs for such cases.

A Plethora of other features



kernel

A set of programs that form from the heart of OS.

Handles key responsibilities such as managing processes, memory, devices and the file system.

UI

Pains with I/O devices.

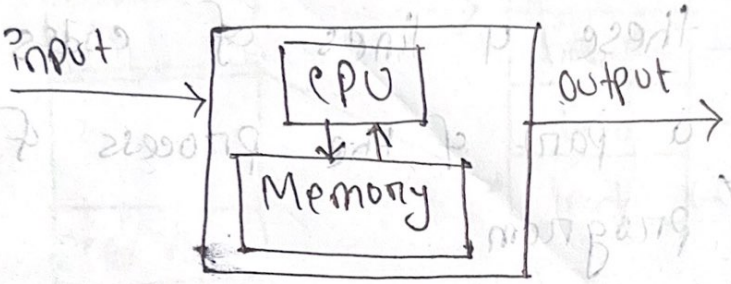
Files Systems

manage files and folders.

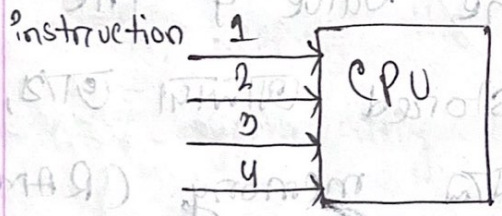
CSE 321 Date: 22/06/25

Operating Systems

Process



Von Neumann Model.



↳ doesn't know why the instruction are being followed.

↳ one task is very different to another task.

↳ OS make the instruction a package, which has all the necessary infonmations. Task 1 Task 2 it is process.

Things that are needed to run a program.

```
a = 10
b = 20
c = a + b
print(c)
```

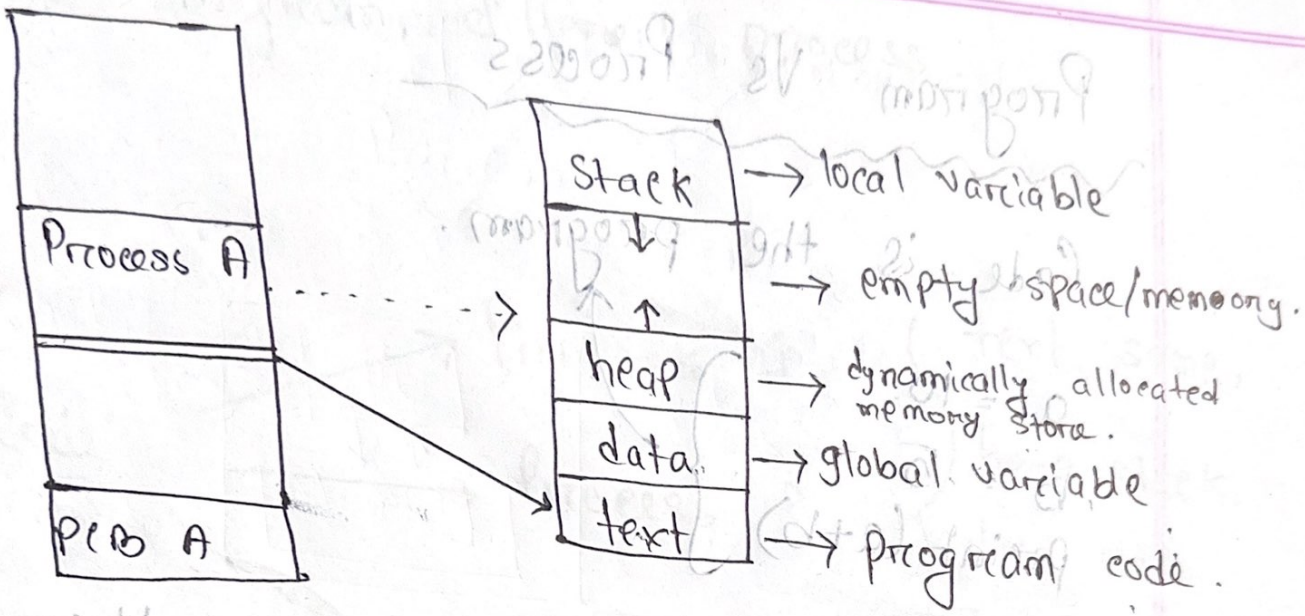
these 4 lines of codes are a part of the process of this program.

Here, we don't store the values.

we keep the value of the variable stored আমরা ডাটা, যেটা আমরা memory (RAM) এ রাখি,

Ram → code base
variable value.

Instruction + Information + code = Package/process



Program start করার পর, extra memory চাইলে, keeping the program running is known as, dynamically allocated memory.

↓ → heap and stack space can be increased or decreased.

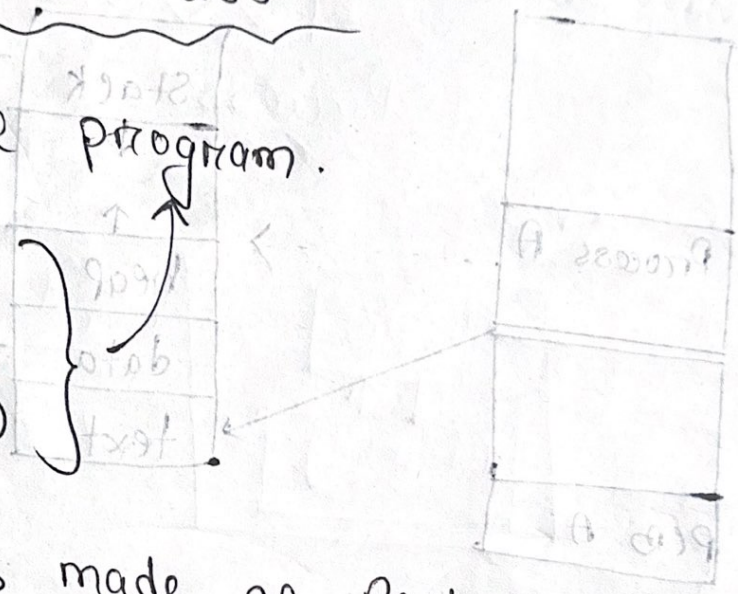
↑ dynamically allocated memory.

↑ recursive function.

Program Vs Process

Code is the program.

```
a = 10
b = 5
Print (a+b)
```

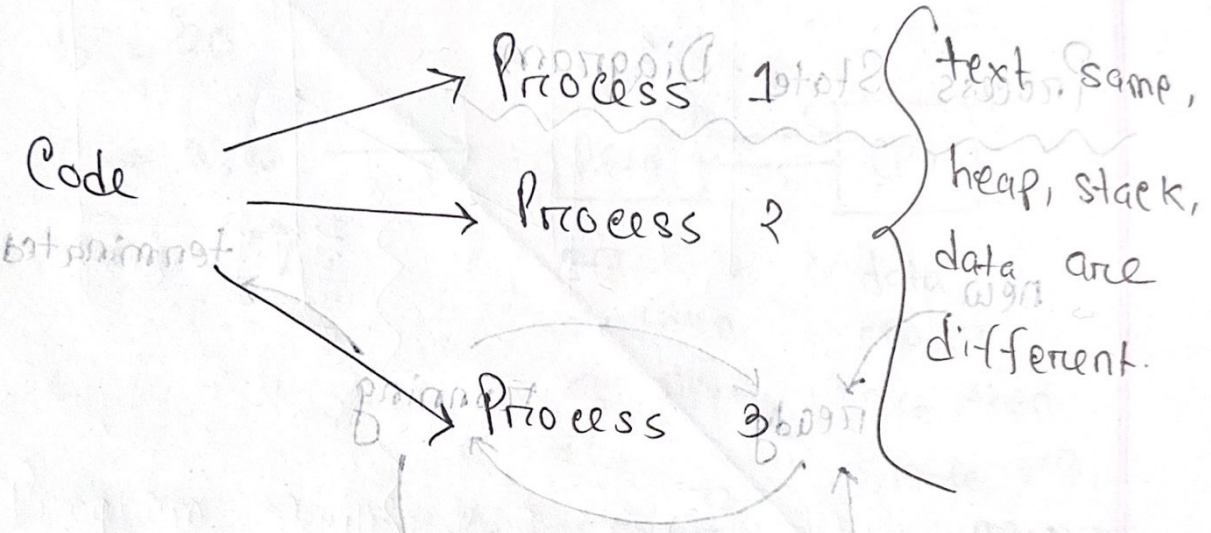


When code is made as package then, it is called a process.

One program (এর) multiple process হতে পারে।

One program এর ↑process এর text same থাকবে but, heap, stack, data ↑ different different থাকবে।

Same program, different process



State of a Process

Process এখানো বানানো হয় নাই, বানানো হচ্ছে,

The states a process can be:

→ New: Process is being created.

→ Running: instructions are being executed.

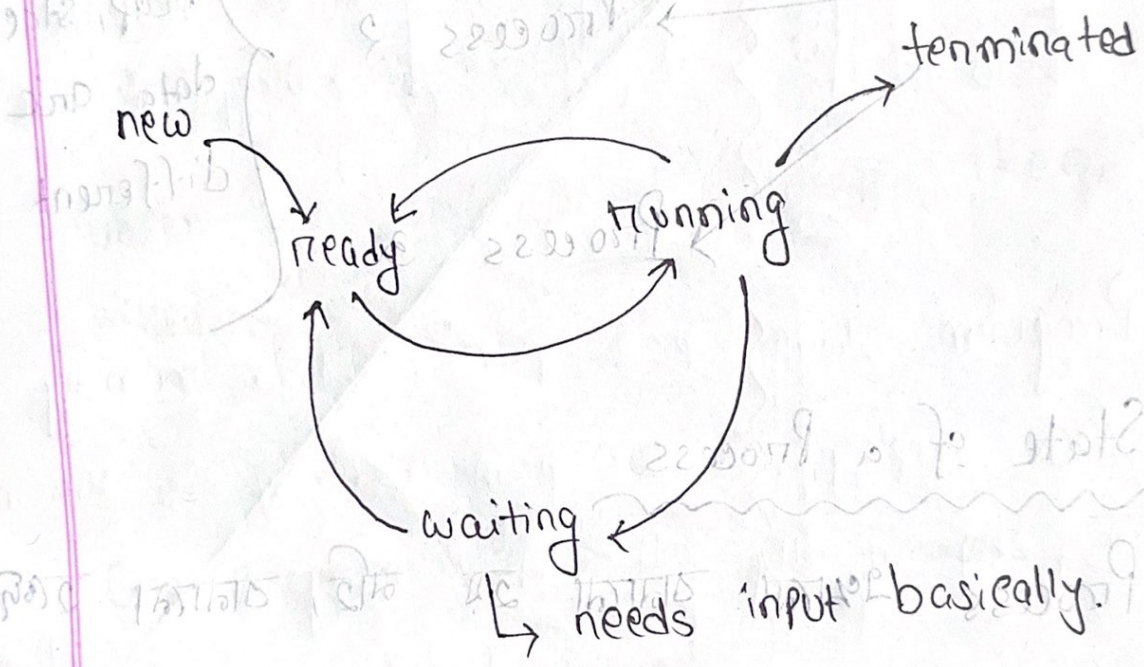
→ Waiting: process is waiting for some event to occur. {instruction needed}.

→ Ready: Waiting to be assigned to a processor. {time, to execute needed}.

Paused এই 2টা but for different reasons.

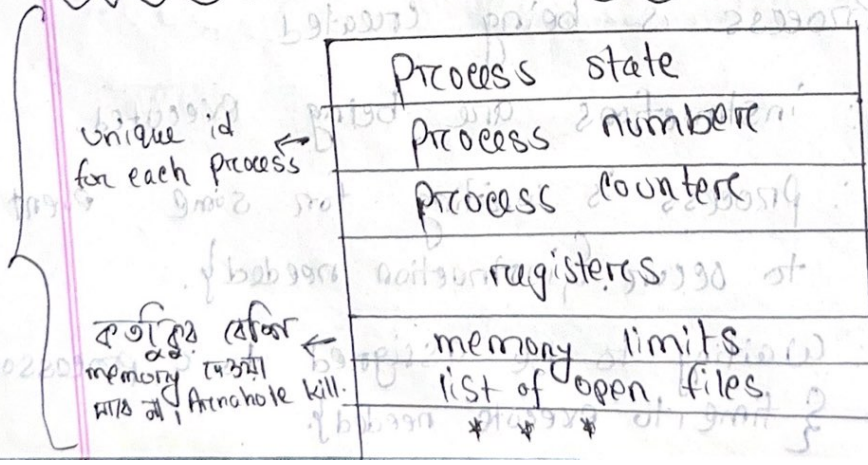
→ Terminated: Process has finished execution.

Process State Diagram



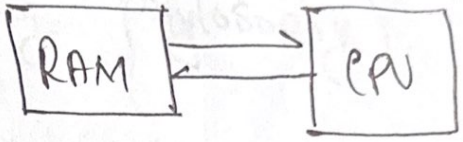
Representation of Process in OS

Process control block (PCB)



registers.

```
a = 10
b = 20
c = a+b
print (c).
```



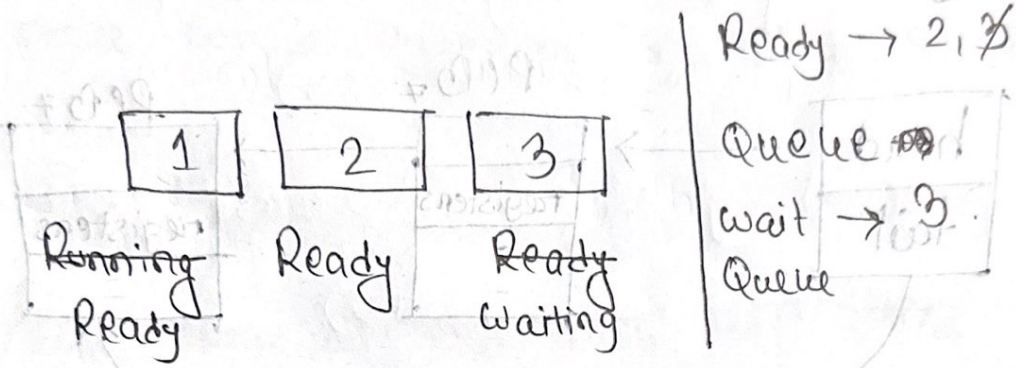
কোন calculation করা যায় না।

data CPU তে গেলে then execute করে, then RAM এ পাঠাবে।

যত info/data, ram থেকে আসে, register এ থাকে, execute করে ও register এ থাকে, Then RAM এ পাঠাও হয়।

CSE 321 Date: 24/06/25

Process Scheduling



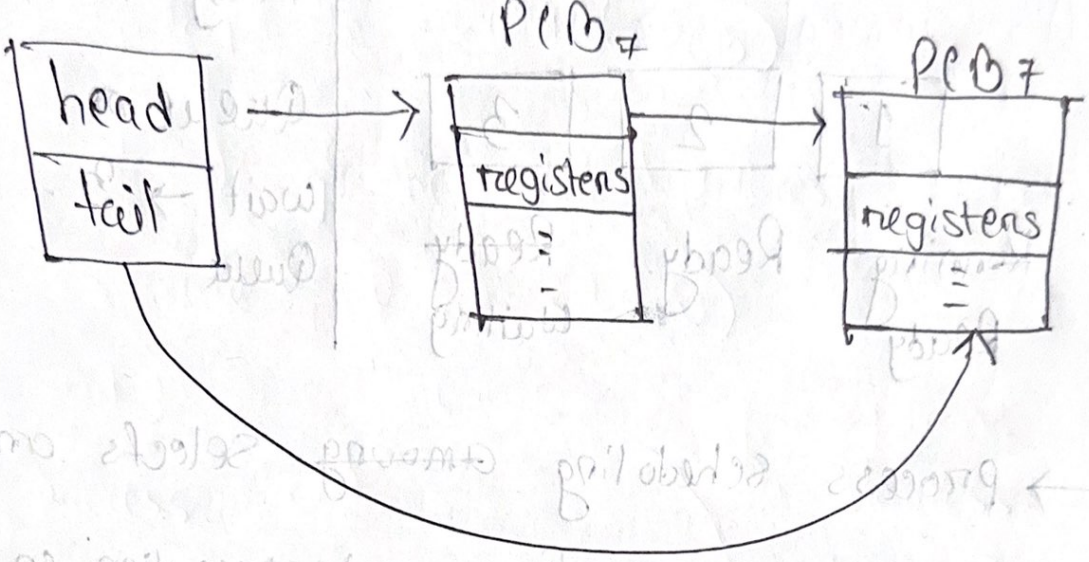
→ process schedoling ~~among~~ selects among available process for next execution on CPU core.

→ Goal.. Maximize CPU use, quickly switch processes onto CPU core.

After the first process, which process will run and for how long time is called process scheduling.

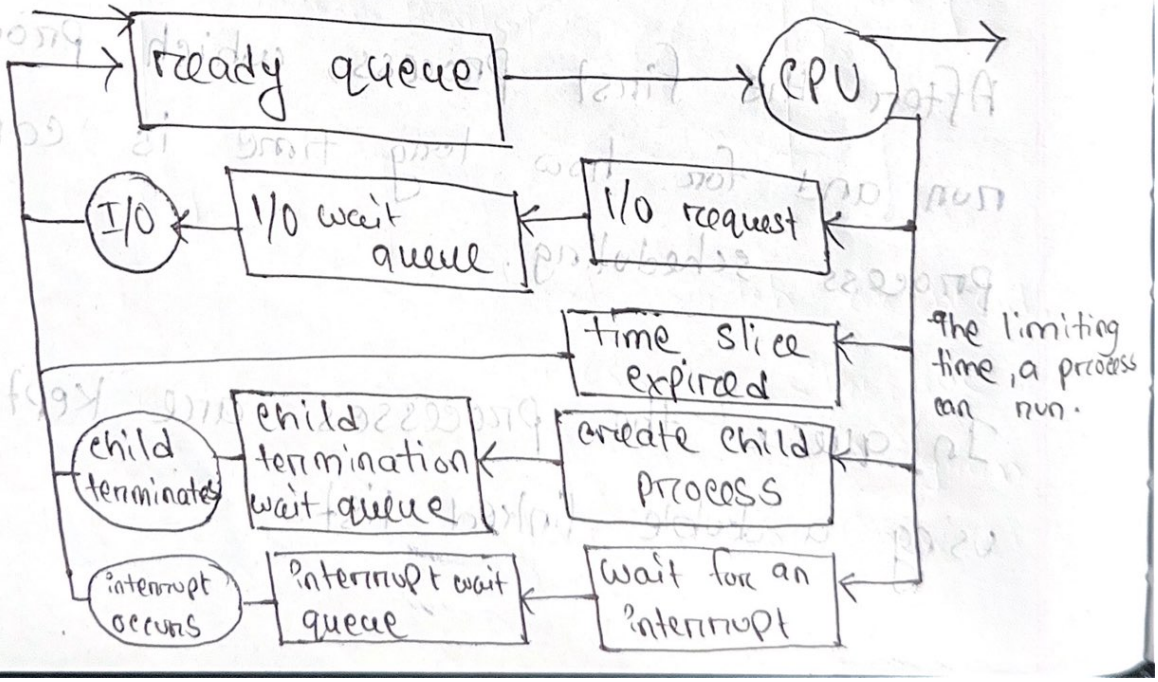
In queue the processes are kept in using a double linked list.

ready queue.



wait queue is same as above.

Representation of Process Scheduling



OS ৯ terminal Alt F4 দিয়ে off করলে notice outside program is called signal / interrupt.

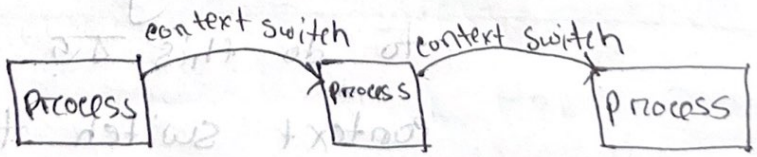
OS ৯ terminal exit দিয়ে off করলে যা inside process

interrupt queue ৯ থাকবে সেই process যাবে, forcefully বন্ধ করা হবে।

Sometimes, interrupt is good.

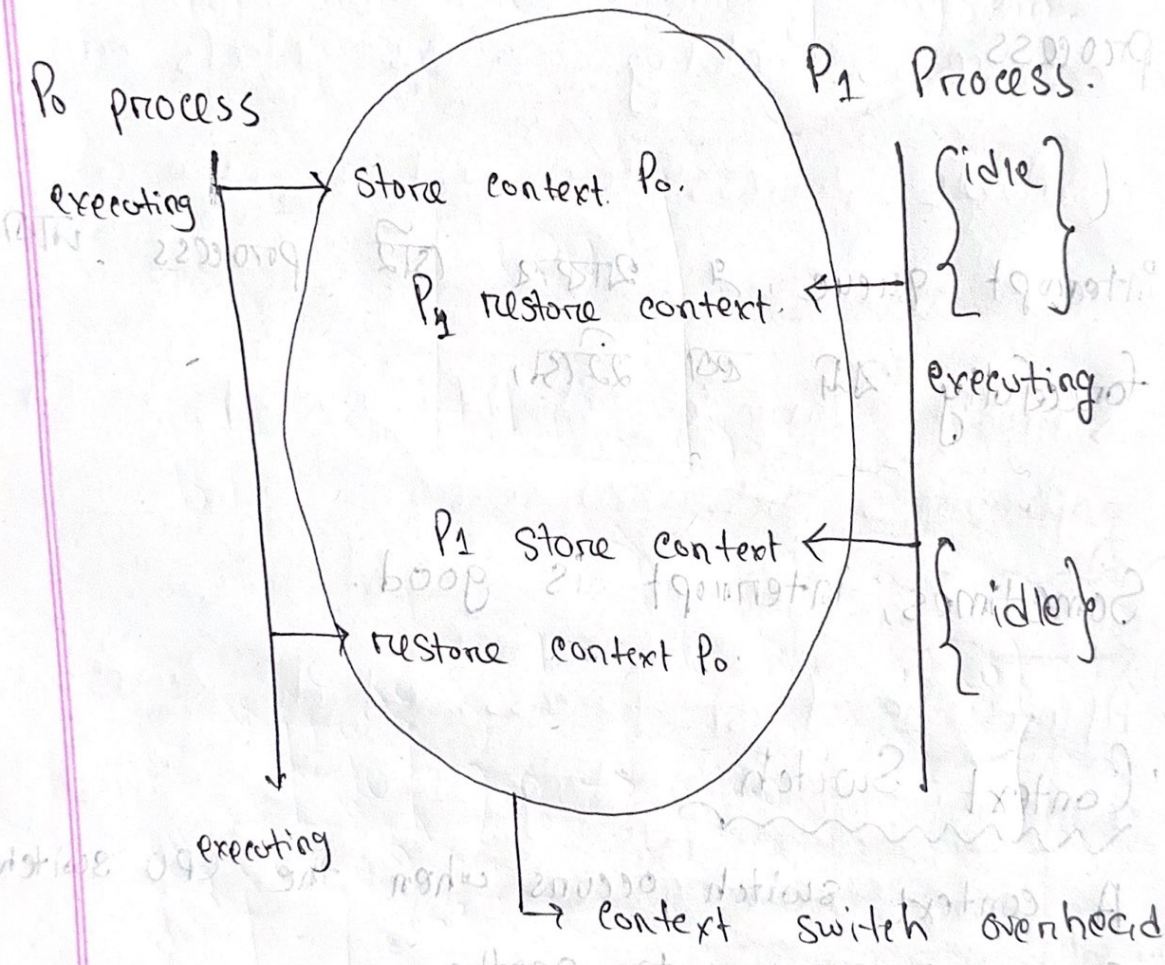
Context Switch

A context switch occurs when the CPU switches from one process to another.



→ Storing currently executed process context in RAM (PCB).

→ Restoring the next process context to execute. (Ram → CPU).



↳ context switch overhead; unnecessary काम, we need to do this. ये program, so, context switch time आता।

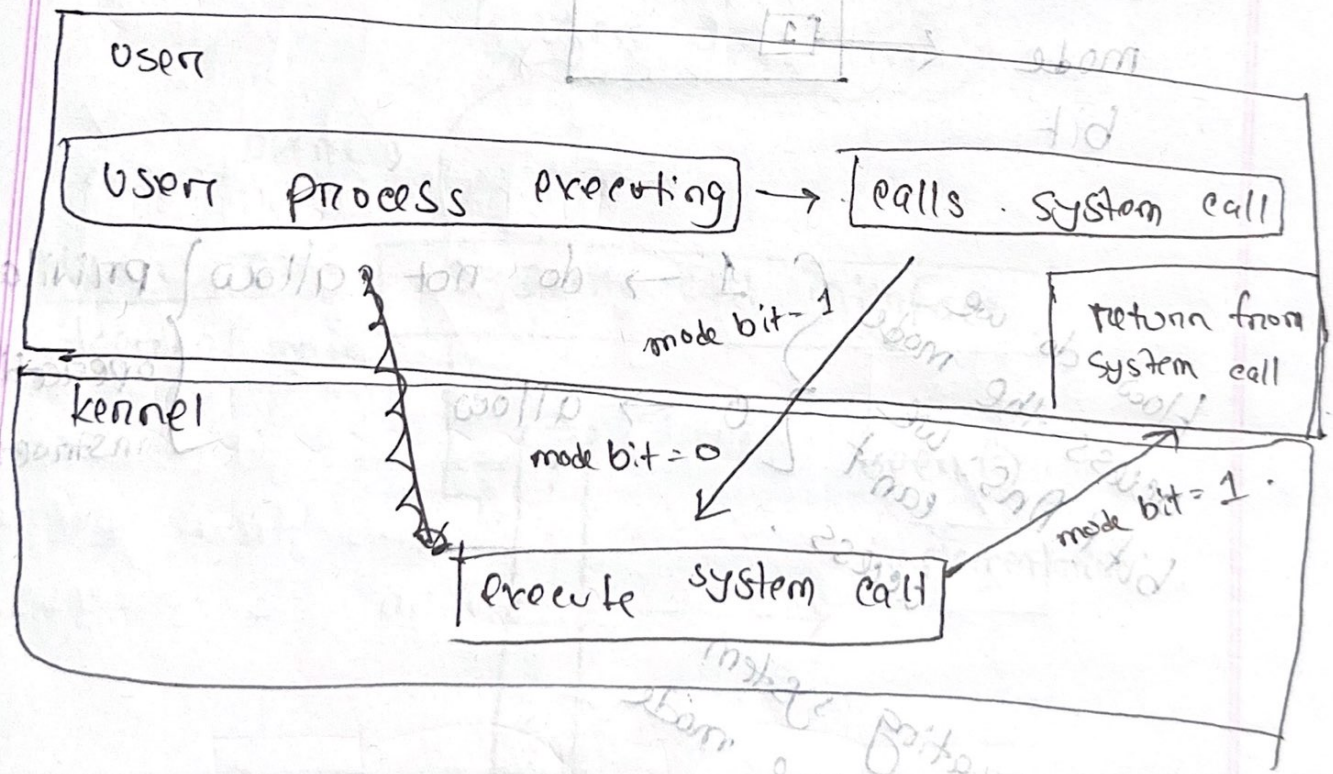
* Python নিজে কোনি file delete করতে পারে না।

Python OS এই কাজে request করে,

```
Path = input()
OS.remove (Path)
```

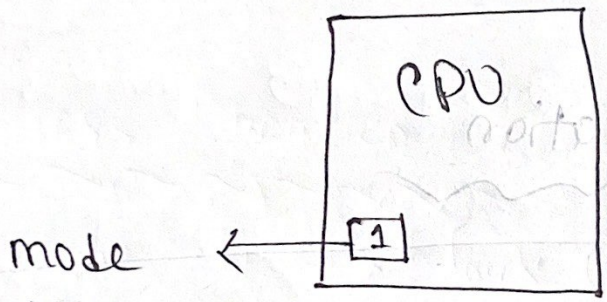
↳ system call.

Dual Mode Operation



CPU does not let to perform the privilaged or instructions.

because if we can perform the privilaged or instructions then, it will be more powerful than the Operating system.



How do we access the mode bit? Ans: we can't access.

- 1 → do not allow → privilaged operations/ instructions.
- 0 → allow → privilaged operations/ instructions.

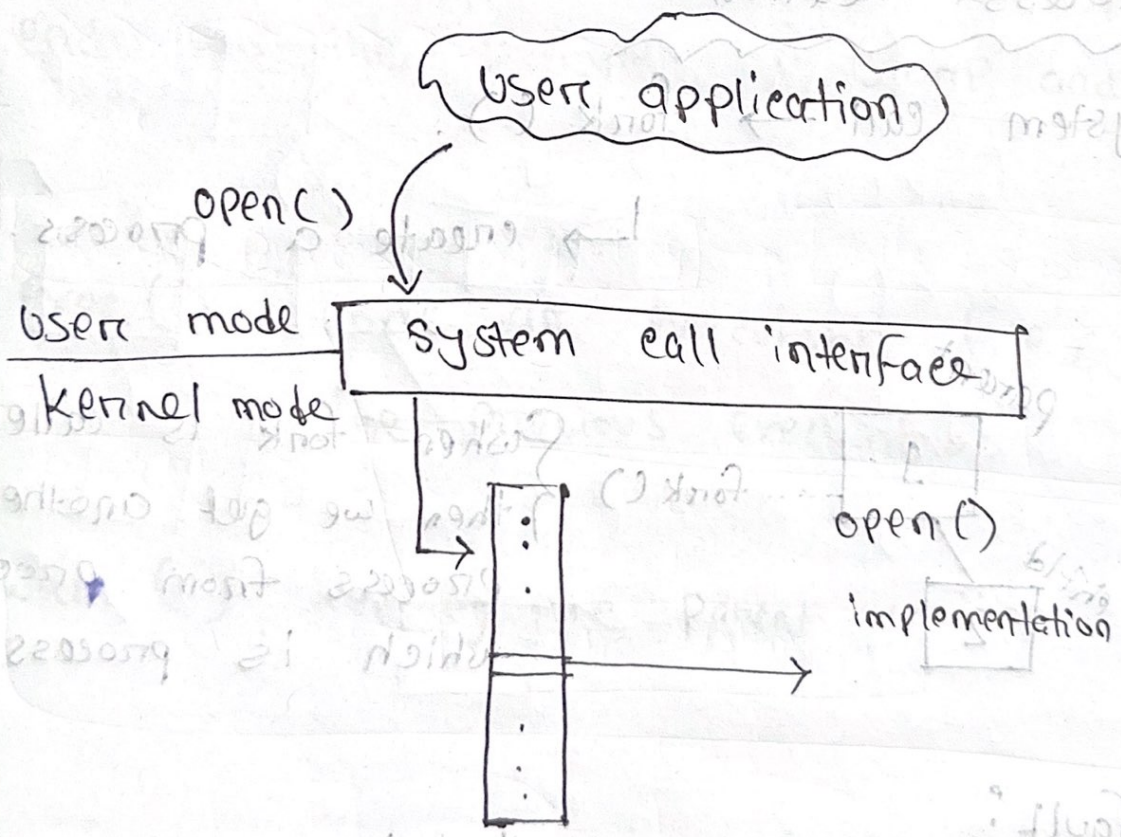
Only operating system can change the mode bit.

System Calls

is a request to the OS.

actually it is a fonction. that acts as the request to the OS.

System call implementation



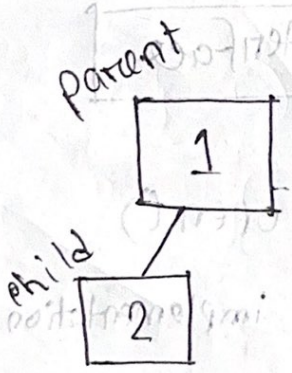
Types of System Call

Lab.

Process Creation

system call → fork().

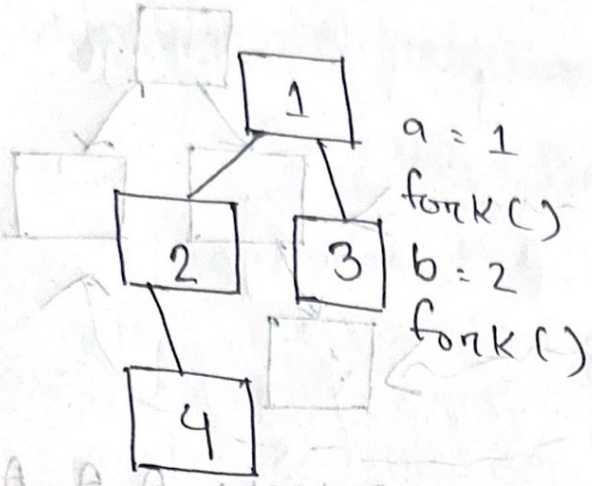
↳ create a process.



when fork is called, then we get another process from process 1 which is process 2.

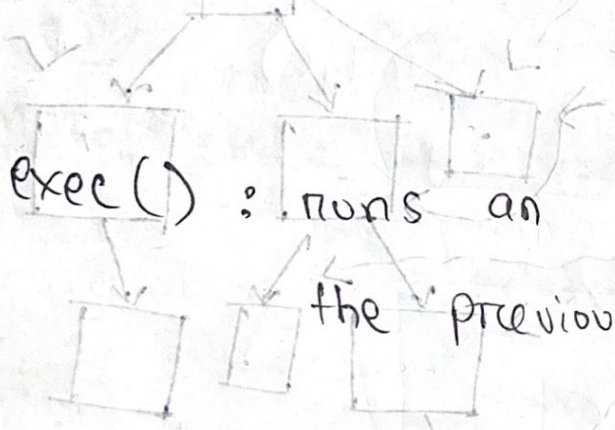
default:

everything is replicated except process id



everything might not be replicated.

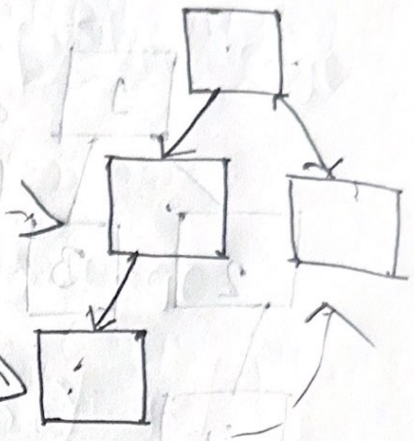
we can run child and parent simultaneously.
or we can choose ~~who~~ ~~will~~ that child will run first then ends and the parent will start and end.



exec() : runs an executable file, replacing the previous executable.

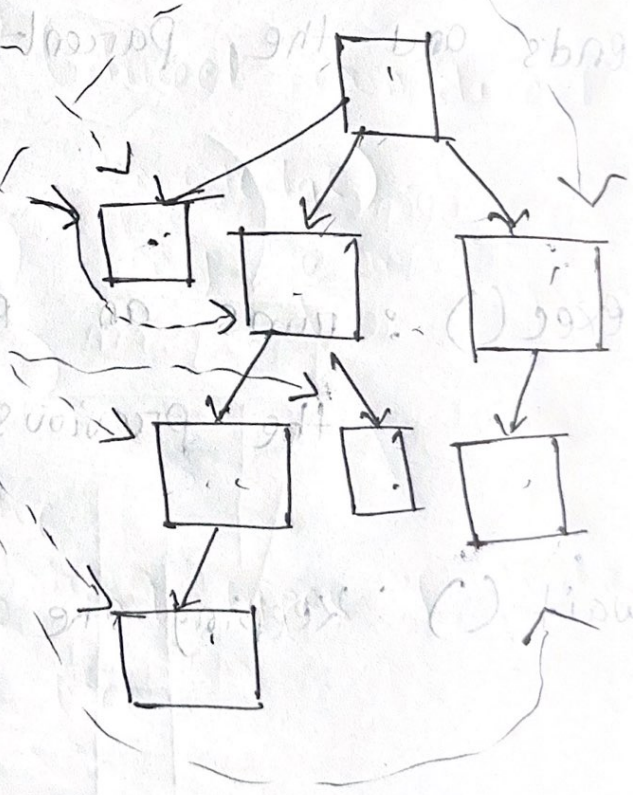
wait() : keeping the parent waiting.

```
int main() {
    fork();
    fork();
    printf("A");
}
```



Output: A A A A

```
int main() {
    fork();
    fork();
    fork();
    printf("A");
}
```



2^n = Process.

= A A A A A A A A.

CSE 321

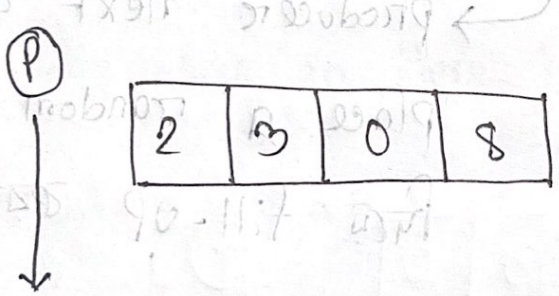
Date: 01/07/25

Shared memory System

Producer - consumer problem

Producer → Produces products for consumer

Consumer → consumes products provided by producer.



fill array with integers until array is full.

read elements one by one and remove from array. until array is empty.

Producer and consumer is done simultaneously.