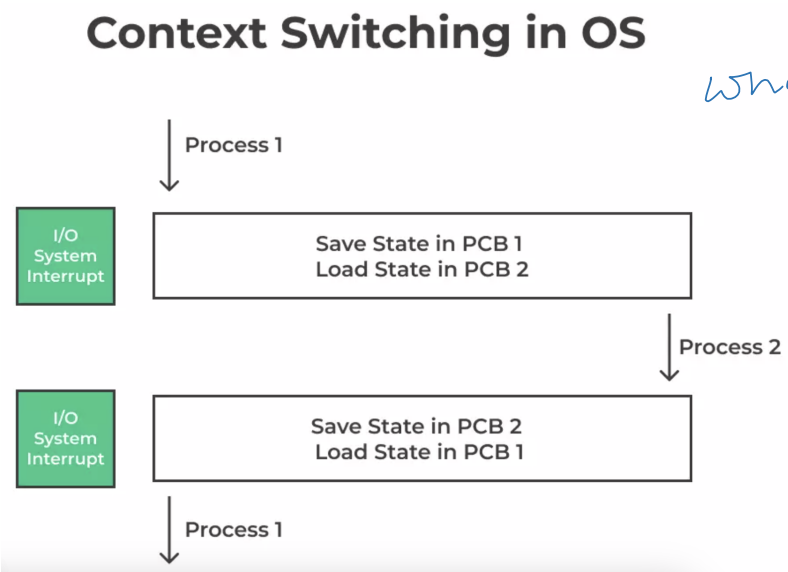
OS2 - CPU Scheduling and Threads

Tuesday, 27 June 2023 12:28 PM

Context Switching

Context: process related data - variables etc.



PCB process control Block

when PCB changes from one process to another, it has to change whole context of the process including the current state of the process

That's a lot of overhead work for switching processes. Degrades performance

Here we are going to discuss some stuff within following constraints:

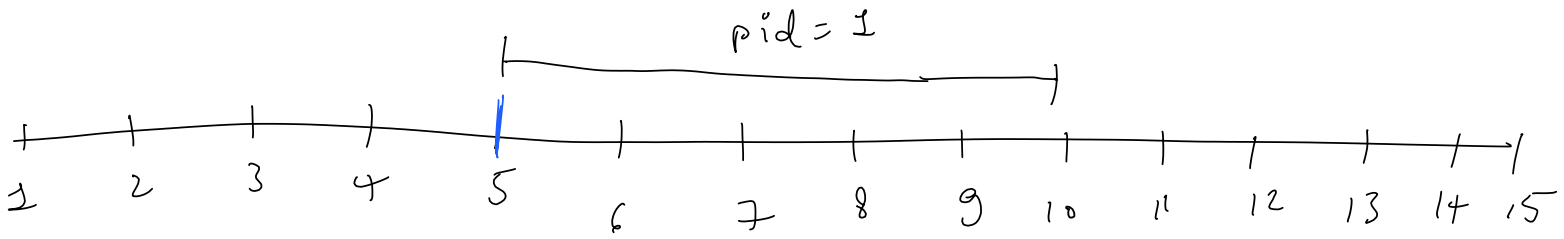
1. Context Switch Time = 0
2. CPU Bound Processes
3. CPU is Single Core

Scheduling Algorithm

1. First Come First Serve

pid	arrival time	Burst time
1	5	5
2	7	3
3	10	2

→ how much CPU time process needs to finish



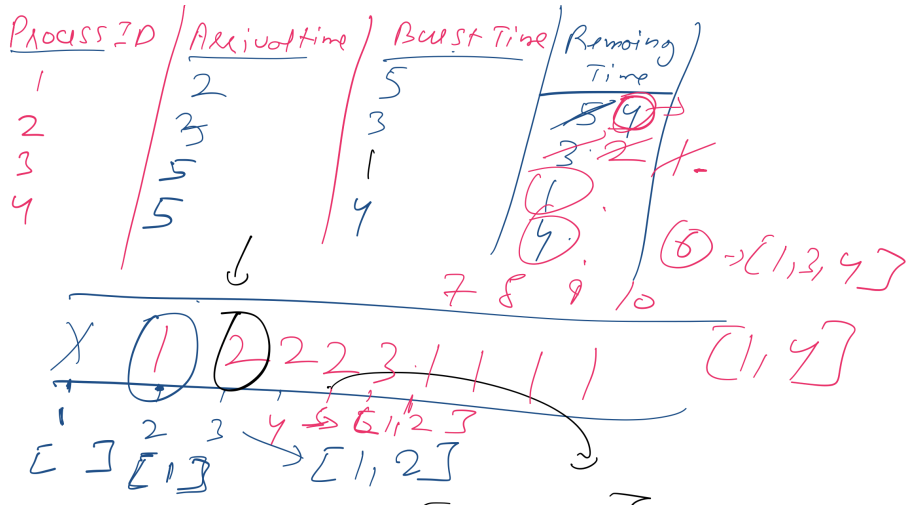
[1] [2] [2,3]

↳ which one to pick?

FCFS says : go with one with lesser arrival time

FCFS is Non-Preemptive

(2) Shortest Remaining Time first [SRTF]



So an ongoing process is stopped if another process comes that has a shorter remaining time. This makes this algo a pre-emptive algorithm.

But in case there are two processes (either both pending or one pending and one ongoing) that have same remaining time i.e. when a clash occurs - priority is given to smaller process id (pid).

(3) Shortest Job First

↳ Non-preemptive version of SRTF

↳ lesser burst time is prioritised while scheduling

Starvation: when a process is not able to complete due to other processes.

this is a problem in all 3 algos discussed above.

Round Robin Scheduling

It has two important data structures:
1. Queue - to store processes
2. Time Quantum Q - an int denoting max cpu time a process can take

So any process is only executed for a max of Q units of time. Then it is dequed from front of the queue and enqued at the back. The next process is then executed.
If a process takes less than Q time to execute, it executes and pops off the queue.
Any new process is always enqueued to the back.

This algo is most generally used in Load Balancers.

Read about Priority Based Scheduling

Throughput → No. of processes a processor can execute in a unit of time.

Latency → Average time it takes a process to complete from the time CPU first schedules it.

This is an average time of all the processes' individual latency times.
The time it takes each process to finish since it was first known to the CPU divided by total no. of processes.

Assignment: Latency Comparison of SRTF vs RR

Throughput :: SRTV vs RR

Threads

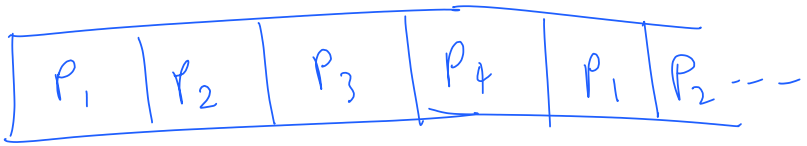
MS WORD:

P_1 = UI

P_2 = auto-suggestion

P_3 = spell-checker

P_4 = updation



MS-word is broken into several processes and each process is executed in Round Robin or something similar.

In order to run the P2 to give auto-suggestion, other processes are stopped - even P1 for UI. The UI freezes but for such a small duration that it is not discernible.

So, such Single Core Systems create an Illusion of multi-tasking where in reality they are just round-robinning the shit out of processes.

Let's say there is a humungous process and it has many sub-processes. Now there can be certain resources like variables that are common to multiple procsses. But inside the PCB of each of the processes, the same data is loaded again and again which creates a memory overhead and it impacts performance.

Thread is the actual basic unit of execution of a CPU. Every process is broken down into several threads and each thread is then executed individually.

For a program, a Main thread is created and then further threads are created on top of it, that share the same memory