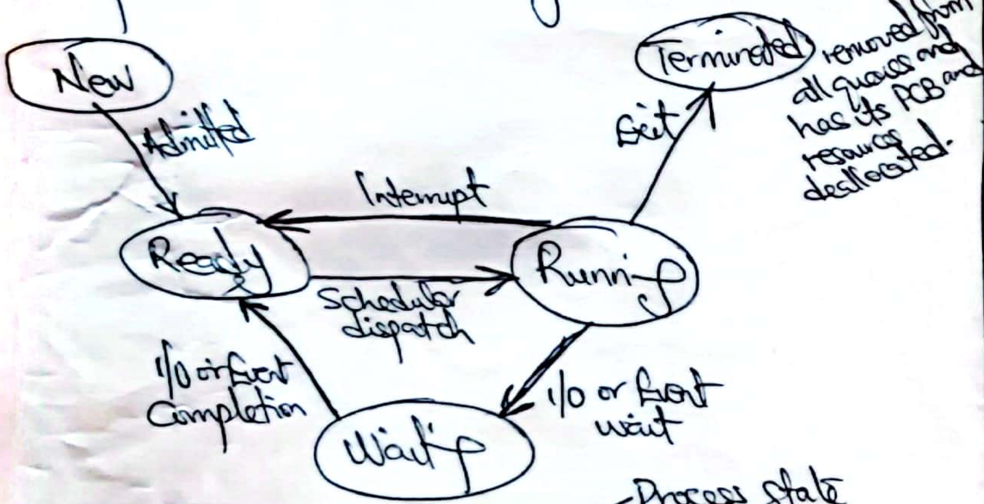
Process

- A program in execution
- While a pgm is just a passive collection of instructions. A process is the actual execution of those instructions.
- A process consists of the ffg resources:
  - (i) Memory
  - (ii) Operating system descriptors of resources that are allocated to the process
  - (iii) Security attributes
  - (iv) Processor state

Process states

As a process executes, it changes states:



PCB [Process Control Block]

- Process state
- Pgm counter
- CPU registers
- CPU scheduling info
- Memory Mgt info
- Accounting info
- I/O status info

Scheduling Queues

- As processes enter the system, they are put into a job queue
- Ready processes waiting to be executed are kept in a list called the ready queue; stored as linked list.
- The list of processes waiting for a particular I/O device is called a device queue.

Schedulers

- schedules processes into queues
- Long Term Scheduler (Job scheduler) — selects processes from a pool of processes and loads them into memory for execution.
- The Short-term Scheduler (CPU scheduler) selects from among the processes that are ready to execute, and allocates the CPU to one of them.
- The primary distinction b/w these 2 schedulers is the frequency of their execution.
- STS schedules more frequently and is faster while LTS executes much less frequently and controls the degree of multipgming i.e the no of processes in memory.

CPU–I/O Burst Cycle

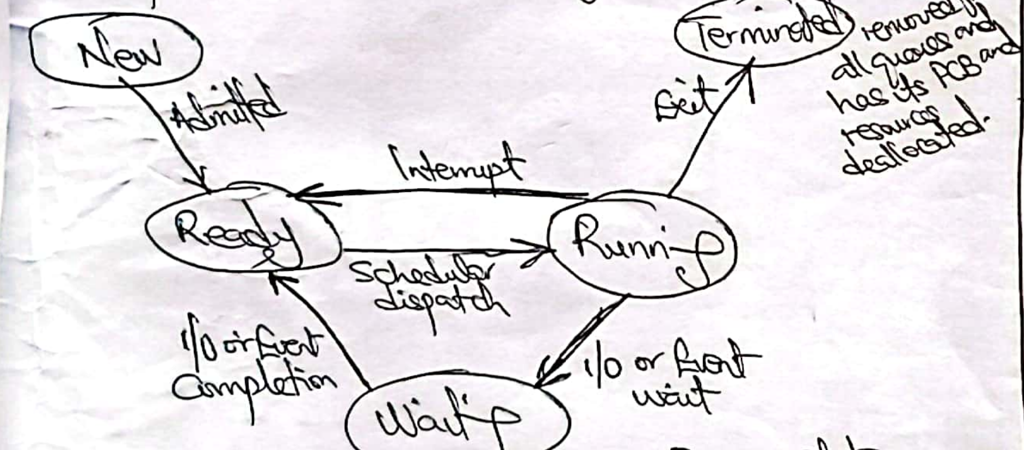
- Process execution consists of a cycle of CPU execution and I/O wait.
- Processes alternate b/w these 2 states.
- Process execution begins with a CPU burst followed by an I/O burst, an so on. The last CPU burst will end c̄ a request to terminate execution.
- CPU burst can be extensively measured.
- An I/O-bound pgm ⇒ has many very short CPU bursts
- CPU-bound pgm ⇒ has a few very long CPU bursts

Process

- A program in execution
- While a pgm is just a passive collection of instructions A process is the actual execution of those instructions.
- A process consists of the ffg resources:
  1. Memory
  2. Operating system descriptors of resources that are allocated to the process
  3. Security attributes
  4. Processor state

Process States

As a process executes, it changes states:



PCB [Process Control Block]

- Process state
- Pgm counter
- CPU registers
- CPU scheduling info
- Memory Mgt Info
- Accounting info
- I/O status info

Scheduling Queues

- As processes enter the system, they are put into a job queue
- Ready Processes waiting to be executed are kept in a list called the ready queue; stored as linked list.
- The list of processes waiting for a particular I/O device is called a device queue.

Schedulers

- schedules processes into queues
- Long Term Scheduler (Job scheduler) selects processes from a pool of processes and loads them into memory for execution.
- The Short-Term Scheduler (CPU scheduler) selects from among the processes that are ready to execute, and allocates the CPU to one of them.
- The primary distinction b/w these 2 schedulers is the frequency of their execution.
- STS schedules more frequently and is faster while LTS executes much less frequently and controls the degree of multiprogramming ie the no of processes in memory.

CPU-I/O Burst Cycle

- Process execution consists of a cycle of CPU execution and I/O wait.
- Processes alternate b/w these 2 states.
- Process execution begins with a CPU burst followed by an I/O burst, and so on. The last CPU burst will end with a request to terminate execution.
- CPU burst can be extensively measured.
- An I/O bound pgm ⇒ has many very short CPU bursts
- CPU-bound pgm ⇒ has a few very long CPU bursts

Preemptive Scheduling & Non-Preemptive Scheduling

Non-Preemptive Scheduling, once CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.

In a preemptive Scheduling the CPU is forcefully taken away from the process.

① when a process switches from the running state to the ready state (eg when interrupts occurs)

② When a process switches from the waiting state to the ready state (Completion of I/O).

Disadvantages of Preemptive Scheduling:

- It is costly
- has effect on the design of OS kernel.

Scheduling Criteria

(i) CPU utilization
(ii) Throughput
(iii) Turnaround Time — Time interval from submission of a process to the time of Completion
(iv) Waiting Time → The amount of time a process spends waiting in the ready queue.
(v) Response Time → Amount of time it takes to start responding but not the time it takes to output the response i.e the time from the submission of a request until the first response is produced.

Scheduling Algorithms

- Deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU.

FCFS Scheduling → the process that requests the CPU first is allocated the CPU first.

- easily managed with a FIFO queue.
- The Average waiting time under this policy is quite long

Process	Burst time
P_1	24
P_2	3
P_3	3

P_1	P_2	P_3
0 – 24	24 – 27	27 – 30

$AWT = (0+24+27)/3 = 17ms.$

Assuming the processes arrive in the order $P_2 P_3 P_1$

P_2	P_3	P_1
0 – 3	3 – 6	6 – 30

$AWT = (6+0+3)/3 = 3$ ms

② SJF Scheduling

- This algorithm associates with each process the length of the CPU burst.
- If 2 processes have same length CPU burst, then use FCFS to break the tie.

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3

P_4	P_1	P_3	P_2
0 – 3	3 – 9	9 – 16	16 – 24

$AWT = (3+16+9+0)/4 = 10.25$ millisecs.

- SJF algorithm is optimal b/c it gives the minimum average waiting time for a given set of processes.
- It cannot be implemented at the level of short-term CPU scheduling b/c there is no way to know the length of the next CPU burst.
- May be either preemptive or non-preemptive.
- Preemptive SJF scheduling is sometimes called shortest-remaining-time-first scheduling.

Consider the ffg 4 processes, with length of the CPU-burst given in milliseconds.

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

If the processes arrive at the ready queue at the times shown, then the Preemptive SJF is shown in Gantt chart below:

P_1	P_2	P_4	P_1	P_3
0–1	1–5	5–10	10–17	17–26

$$AWT = ((10-1)+(1-1)+(17-2)+(5-3))/4$$
$$= 26/4 = 6.5 \text{ millisecs.}$$

A Non-Preemptive SJF scheduling (Assignment) (7.75 millisecs)

3) Priority Scheduling

- CPU is allocated to the process with the highest priority
- Equal priority processes are scheduled in FCFS.
- The higher the CPU burst, the lower the priority & vice versa.
- Low nos ⇒ High priority

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Using priority scheduling, the processes will be scheduled thus:

P_2	P_5	P_1	P_3	P_4
0–1	1–6	6–16	16–18	18–19

$$AWT = (6+0+16+18+1)/5 = 8.2 \text{ milliseconds}$$

Disadvantage ⇒ Indefinite blocking or starvation.

Soln ⇒ Aging (a technique of gradually increasing the priority of processes that wait in the system for a long time).

4) Round-Robin Scheduling

- Simple & easy to implement
- Assigns time slices to each process in equal portions and in circular order, handling all processes without priority.
- Starvation free.
- Similar to FCFS scheduling, but preemption is added to switch b/w processes.
- A small unit of time called a time quantum/time slice is defined. Generally from 10–100 milliseconds.
- To implement this, we keep the ready queue as a FIFO queue of processes.
- The Average waiting time is quite long.

Eg. Consider the ffg set of processes, assumed to have arrived at time 0, in the order P_1, P_2, P_3, with the length of the CPU burst given in milliseconds.

Process	Burst Time
P_1	24
P_2	3
P_3	3

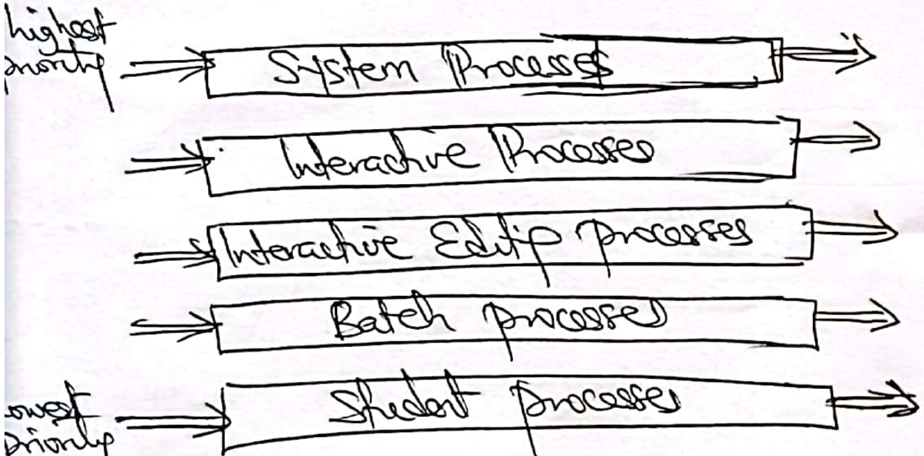
Time Quantum = 5ms

P_1	P_2	P_3	P_1	P_1	P_1	P_1
0–5	5–8	8–11	11–16	16–21	21–26	26–30

$$AWT = ((11-5)+5+8)/3$$
$$= (6+5+8)/3 = 19/3$$
$$= 6.3 \text{ millisecs}$$

5) Multilevel Queue (MLQ) Scheduling

- Processes are classified into different groups.
- The processes are permanently assigned to one queue, generally based on some property of the process → memory size, process priority etc.
- Each queue has its own scheduling algorithm
- E.g. foreground might use RR while background might use FCFS.



Systematic Approach to Performance Evaluation

1. State Goals & Define the System
2. List Services and Outcomes
3. Select Metrics
4. List Parameters
5. Select factors to study
6. Select Evaluation Technique
7. Select Workload
8. Design Experiments
9. Analyze & Interpret Data
10. Present Results

Commonly Used Performance Metrics

1. Response time — The total amount of time it takes to respond to a request for service.
   - Response time is the sum of the service time and wait time.
   - Increases as the load on the system increases.
2. Throughput → the maximum rate of production.
   - In communication networks, network throughput is the rate of successful message delivery over a communication channel.
   - For batch systems, throughput is measured in jobs per second.
   - For interactive systems, it is measured in requests per second.
   - For CPUs, it is measured in Millions of Instructions per second (MIPS) or Millions of Floating Point Operations per second (MFLOPS)
   - For Networks, it is measured in packets per second (pps) or bits per second (bps)
   - For Transaction Processing Systems, it is measured in Transactions per second (TPS)
   - System throughput generally increases as the load on the system initially increases.
3. The maximum achievable throughput under ideal workload conditions is called Nominal Capacity of the system.
   - Nominal capacity ≡ Bandwidth expressed in bits per second in Cpt Networks.

3) Efficiency → This is the ratio of maximum achievable throughput to nominal capacity.
- The ratio of the performance of an n-processor system to ~~that~~ of a one-processor system is its efficiency.
- The performance is usually measured in terms of MIPS or MFLOPS.

4) Utilization → of a resource is measured as the fraction of time the resource is busy servicing requests.

5) Reliability - of a system is usually measured by the probability of errors or by the mean time b/w errors.
- The time during which the system is not available is called Downtime.
- The time during which the system is available is called Uptime.
- Mean Uptime is also known as the Mean Time To Failure (MTTF)

6) Availability - is a xtic of a system, which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.

7) Transmission Time - gets added to ~~network~~ response time when your request and the resulting response has to travel over a network and it can be very significant.
- This includes propagation delays due to distance, delays due to transmission errors, and data communication bandwidth limits

8)

Amdahl's Law

- is an expression used to find the maximum expected improvement to an overall system when only part of the system is improved.
- often used in parallel computing to predict the theoretical maximum speedup using multiple processors.
- Amdahl's law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.
- Amdahl's law is formulated thus:

$$S_{latency}(S) = \frac{1}{(1-P)+\frac{P}{S}}$$

where; $S_{latency}(S)$ is the theoretical speedup of the execution of the whole task;
- S is the speedup of the part of the task that benefits from improved system resources
- P is the proportion of execution time that the part benefitting from improved resources originally occupied

Furthermore, $S_{latency}(S) \leq \frac{1}{1-P}$

$$\lim_{S \to \infty} S_{latency}(S) = \frac{1}{1-P}$$

Processor Performance Eqn

CPU time = CPU clock cycles for a program × clock cycle time

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{clock rate}}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction Count}}$$

Cycles per instruction

CPU time = Instruction count × Cycles per instruction × clock cycle time

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{clock cycle}} = \frac{\text{Seconds}}{\text{Program}}$$

$$\text{Performance}(X) = \frac{1}{\text{Execution-time}(X)}$$

"X is n times faster than Y" means

$$n = \frac{\text{Performance}(X)}{\text{Performance}(Y)} = \frac{\text{Execution-time}(Y)}{\text{Execution time}(X)}$$

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

$$\text{Speedup} = \frac{\text{Execution time b/f Improvement}}{\text{Execution time after improvement}}$$

* If a cpt A runs a pgm in 10 secs and cpt B runs the same pgm in 15 seconds, which cpt is faster? and How much faster?

Ans A is n times faster than B if

$$\frac{\text{Performance of A}}{\text{Performance of B}} = \frac{\text{Execution-time of B}}{\text{Execution time of A}} = n$$

The performance ratio n is $\frac{15}{10} = 1.5$

So A is 1.5 times (50%) faster than B.