Threads

Process

Def

Context switch

Creation

Termination

Execution

Current State
- Pointer

Event info.
waiting process
List of open files

Heap
Process States
- new state → process creation.
- ready state → ready to run but waiting for CPU
- executing state → CPU allocates memory
- waiting state

Process ID

Priority (Scheduling Queue)

PC

Register
- GPR - IR - SP - Acc.

Accounting
- user & Kernel CPU time consumed

Memory allocation.

User level
- implemented by User
- OS doesn't recognize
- easy
- CS - less
- No hardware
#block → whole process blocked

Kernel level
- implemented by O.S.
- O.S recognize
- complex
- CS - more
- Hardware
- if blocked, another will get executed

Process
Context switch
Process States, Description.
PC, structure
PCB
Operations on Process

Operation on Process
Process Creation
- System initialization.
- I R P E P C S C
- A U R C N P
- Starting a Batch Job

Process termination.
EXIT

Threads, types of Threads.
Multithreading
Multicore processing
Amdahl's law
Schedulers, types of schedulers.

One to One
- more concurrency & parallel
- Kernel → User → API degrade

AMDAHL'S LAW

speed up = time required to execute a pg. on single process / time required to execute a pg. on no. of process = 1 / (1-F) + F/N

Many to one
- management done in user space
- if block

Scheduling Long.
Select process → queue
↳ loads in memory for exec.
- speed less
- new ready
- NU
- Job Sched.

Medium
- swap in & out from memory
- speed is in between
- new state
MTA.

Short
select process from R.Q & assign it to CPU
- speed V. fast
- ready to execute state
- CPU sched.

1 to M
- NW of V.LT

Uniprocessor Scheduling

- improve perf. by keeping CPU busy all the time

1) CPU utilization
2) Throughput
3) Turn around time
4) Waiting time
5) Response time
6) Fairness

Non preemptive
- once process is allocated to CPU it does not free CPU until it completes its execution

Preemptive
- it allows taking away CPU from process during execution

Thread Scheduling.

1) User level
2) Kernel level

Multiprocessor Scheduling.

Process Scheduling

- if 1> processor present then scheduling approach.
  ⇐ less imp.

Ways to use thread Scheduling

1) Load Sharing
2) Gang Scheduling
3) Dedicated Process Assignment
4) Dynamic Scheduling

Linux Scheduling →

RT	T

Scheduling Algorithms

1) First in First Out (FIFO) → NP
- allocates the CPU to process in order of their arrival
- Short job has wait until execution of long job (1)

2) Shortest Job First (SJF) → NP/P
- reordering the jobs so as to run SJF → improves response time
- minimize the average waiting time

3) Priority Scheduling
- each process has priority → int value assigned to it
- smallest int → highest P.
- largest int → lowest P.

4) Round Robin Scheduling.
- time sharing system
- many processes get CPU on time sharing basis
- smallest unit time → quantum

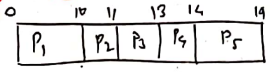
5) Multilevel queue Scheduling.
- categorites the process into ≠ groups.
- separation between interactive process & batch process.

6) Multilevel Feedback-Queue Scheduling.
- process are not allowed to transfer from one queue to another
- Queue → permanent & cannot be change
- low scheduling cost
- not flexible

Numericals

Process	BT	AT	P
P_1	10	0	3
P_2	1	0	1
P_3	2	0	3
P_4	1	0	4
P_5	5	0	2

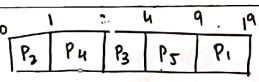
(i) FCFS



Gantt chart

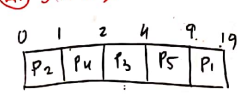
Process	P	FT	TAT	WT
P_1	3	10	10	0
P_2	1	11	11	10
P_3	3	13	13	11
P_4	4	14	14	13
P_5	2	19	19	14

(ii) SJF (NP)



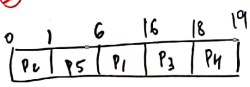
Process	P	FT	TAT	WT = FR-A
P_1	3	19	19	9
P_2	1	1	1	0
P_3	3	4	4	2
P_4	4	2	2	1
P_5	2	9	9	4

(iii) STF(P)



Process	P	FT	TAT	WT
P_1	3	19	19	9
P_2	1	1	1	0
P_3	3	4	4	2
P_4	4	2	2	1
P_5	2	9	9	4

(iv) Priority Sched. (NP)



Process	P	FT	TAT	WT
P_1	3	16	16	6
P_2	1	1	1	0
P_3	3	18	18	16
P_4	4	19	19	18
P_5	2	6	6	1

(v) Round Robin Time Quantum = 1

Process	P	FT	TAT	WT
P_1	3	19	19	9
P_2	1	2	2	1
P_3	2	7	7	5
P_4	4	4	4	3
P_5	2	14	14	9

$WT(P_1)$
$= (0-0)+(5-1)+(8-6)+(10-9)+(12-11)+(14-13)$
$= 9$