Process Concept →

Programme → Set of Instructions in a sequence for particular task.
↓
Executable file stored in stored memory.

↳ for execution prog. load in one Primary memory.

↳ then executing state is called Process.

Q. Notepad file

↳ Process having own address space.

Address space contains → machine language code.

↳ data segment (global, static, external & other)

↳ Stack segment. → store to temp. data.

↳ PCB → Process control Block → store some info. like → Process id
→ Registers.
→ Prog. counter
→ Process state
↳ etc.

↳ Data structure to maintain OS

from where next address comes

→ Process needs resources → CPU, I/O device
→ memory, Files etc. } to perform task.

→ Types →

User Process

* Execute code, utility or application

System Process

→ Execute OS process
→ these are critical process.
→ functions → Process management
→ memory "
→ I/O "

Parent Process.

→ Process can create — subprocess called child Process

→ child Process can use → All resources of their Parent & others too.

→ ↳ can also create child Process.

→ Once Parent Process terminated → all child Process will also terminated.

Process State: → As a process executes, it changes states.

→ The state of a process is defined in part by the current activity of that process.

New → the process is being created.

Running - Instructions are being executed.

Waiting - the process is waiting for some event to occur (such as an I/O completion or reception of a signal).

Ready → the process is waiting to be assigned to a processor.

Terminated → the process has finished execution.

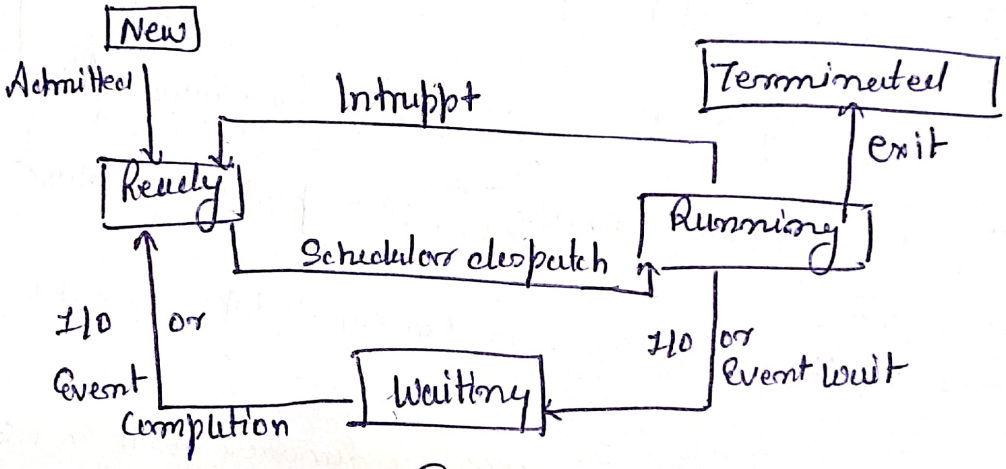


Fig: of Process State

Process Control Block: or **task Control Block:** Each process is represented in the OS by a PCB.

Process state	→ Particular state is in current event.
Process number	→ Unique id or Proc. id.
Programme counter	→ Address of next Instruction will process executed.
Register	→ Used by Process.
Memory Limit	→ memory mgt Information
List of open files	
CPU scheduling	
Accounting Info.	→ Resources used by Particular Process.
I/O status Info.	→ which I/O devices are assigned to " ".

Process Scheduling → objective of multiprog.. to maximize CPU scheduling,

→ objective of time sharing is to switch the CPU to interact with each prog.

→ To meet these two objectives Process scheduler comes in Picture.

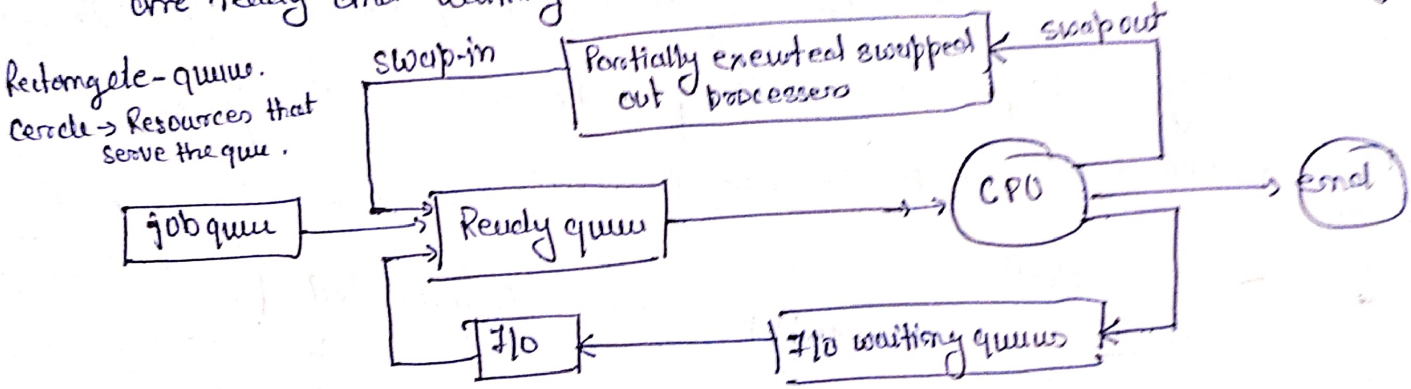
Scheduling queues.

Job queue: As processes enter the system they are put into a job queue, which consists of all processes in the system.

↳ they are not executing.

↳ list of all processes which are waiting to go to ready queue.

Ready queue: The processes that are residing in main memory and are ready and waiting to execute are kept on the list called ready queue.



Device queue:- list of Process waiting for a particular devices is called a device queue.

→ Each device has its own device queue (magnetic tape, disk, I/O devices etc).

Process Scheduling → Particular Process or function with in OS.

→ types.

↳ long term scheduler → from new state – ready state

↳ two ways → (1) Initiated by User.

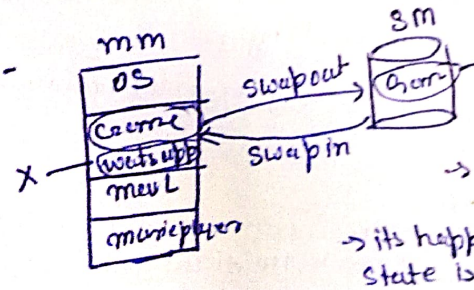
(2). by OS.

↳ short term Scheduler → selects one of all ready process to run.

→ its called also CPU scheduling.

↳ mid term Scheduler:-

↳ Swapping is based on process priority then its called rolling.



→ this state is called suspended.

→ its happend when Process current state is waiting or Ready only.

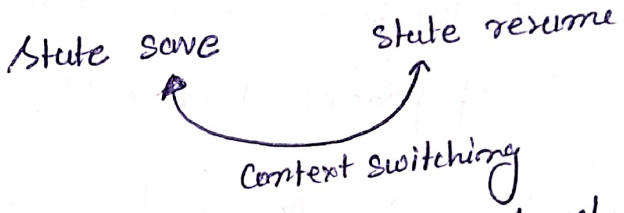
↳ its called suspended Ready or. suspended blak.

Context Switching → Intruppt cause the OS to chenge a CPU from its current task and to run a kernel routine.

→ such operation happen frequently on General purpose system.

→ when Intruppt occurs system save current content of process currently running on the CPU so that it can restore when intruppt process processing is done, essentially suspending the process and then running it.

→ the Context is represented in the PCB of the process.



Note: (1). context switching time is pure overhead.
(2) speed in milli second.
(3) its varied on machine to machine.

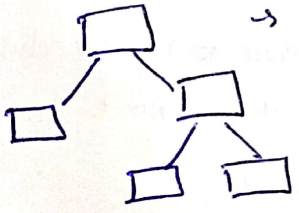
Operation On Processes.

Process Creation! → Process can create new process by create-process system call.

→ creating Process → Parent process
new " → child of that process.

→ children process also create sub process & forming a tree of process

→ when the Process creates a new process two possibilities exists in terms of execution



(1). Parent concurrently executes with its children.
(2). Parent waits untill some or all of its children have terminated.

→ In terms of address space two possibilities are there-
(1). CP is duplicate of Parent proc.
(2). CP has a new prog. loaded into it.

Process termination → → when process finisher its final statement and asks the OS to delete it by using the exit() system call.

→ at that time process may return a status value (Integer) to its parent process (via. wait() system call)

→ resources like memory (Physical & Vertual), open files, I/O buffer are deallocated by buffer.

→ termination can occure other circumantances.

→ only Parent can kill theire children Process.

reasons → (1) child has exceeds its usage of resources.

(2). the task assigned to that child is no longer required.

(3). if parent terminates itself.

Process Identification Information → already read.

Process address space → already read.

CPU Scheduling → Swiching the CPU among process. → Single CPU / multiple CPU.

CPU - I/O. Burst Cycle → Process execution consists of a cycle of CPU execution & I/O wait.

→ Process alternate b/w these two states.

Process execution begins with a CPU burst to I/O burst to CPU to I/O.

→ CPU Scheduling having two ways → Primitive.
→ Non primitive.

dispatcher → the dispatcher is the module that gives control to the ~~the~~ CPU to the process selected by Short term scheduler.

functions → Switching context
→ Switche to the User mode
→ jumping to the proper location in the user prog. to restart that prog.

dispatch latency. Switching time is called.

→ time taken stop one process & start another processor.

Ready Que → P_1 → which Algo. → CPU → P_1

Scheduling Algorithm

- Preemitive ⟷ Priority / Responsiveness
  - SJF or
    - SRTF → Shortest Remaining time first
    - LRTF → longest " " "
    - Round-robin
    - Priority based
- Non Preemitive
  - FCFS → first come first serve
  - SJF → Shortest Job first
  - LJF → longest " "
  - HRRN → Highest Response Ratio Next
  - multilevel Que.

can be Priority based.

CPU Scheduling

1. Arrival time: The time which process Enter the Ready Que or state.
2. Burst time: (duration) time required by a process to get execute on CPU.
3. Completion time: the time at which Process complete its execution
4. turn around time: {Complete time − Arrival time}
5. waiting time: {Turn around time − Burst time}
6. Response time: {(time of at which Process get CPU first time) − (Arrival time)}

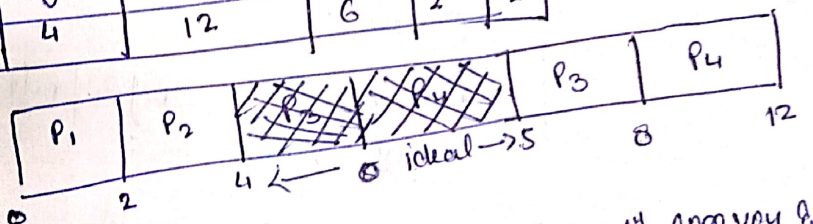
FCFS First come first Serve

Convoy Effect: Smaller Process have to wait for long time for bigger process to release CPU.

Process	(AT) Arrival time	(BT) Burst time	(CT) Completion time	TAT (CT−AT)	WT (TAT−BT)	RT
P_1	0	2	2	2	0	0
P_2	1	2	4	3	1	1
P_3	5	3	8	3	0	0
P_4	6	4	12	6	2	2

Criteria- "Arrival time"
Mode- "Non-Premitive"
AVG TAT − 14/4
AVG WT- 3/4

Gantt chart →

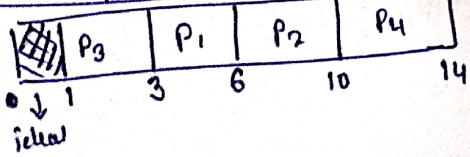


SJF: Shortest Job First → min. BT

Criteria- BT
mode - Non-Prem ⟶ Overcome of convoy Effect

Process	AT	BT	CT	TAT	WT	RT
P_1	1	3	6	5	2	2
P_2	2	4	10	8	4	4
P_3	1	3	3	2	0	0
P_4	4	4	14	10	6	6
AVG			33/4	25/4	12/4	12/4

Gantt chart



(53) SRTF Shortest Remaining time first or SJF for Preemitive

Ex.1

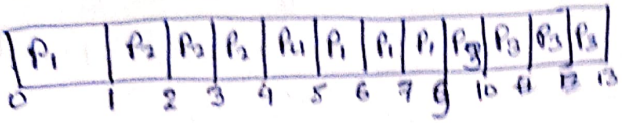
P	AT	BT	CT	TAT	WT	RT
P_1	0	5	9	9	4	0
P_2	1	3	4	3	0	0
P_3	2	4	13	11	7	7
P_4	4	1	5	1	0	0
			6	6	2.75	1.75

avg

Criteria - BT
Mode - Preemitive

↳ Adv:- min. avg time.
↳ ART also better.
disadv → is not possible for implement
↳ starvation is possible.
↳ larger bt have poor RT.

Gnt.

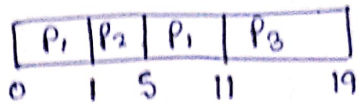


Ex.2

Process	AT	BT
P_1	0	7
P_2	1	4
P_3	2	8

SJF - Preemitive.

Gnt



Round Robin:- Criteria - time Quantum mode - Preemitive.

Pro.	AT	BT	CT	TAT	WT	RT
P_1	0	5	12	12	7	0
P_2	1	4	11	10	6	1
P_3	2	2	6	4	2	2
P_4	4	1	9	5	4	4
			38/4	31/4	19/4	7/4

avg

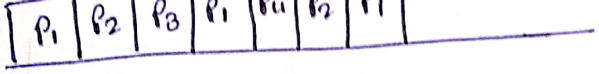
Imp → Sequence of Processes of RIQ.

Given TQ = 2

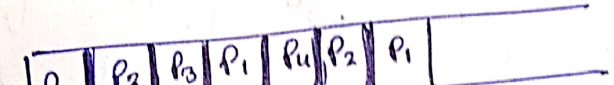
no. of Context Switching. = 6

Adv
↳ Good ART
↳ disadv
↳ Longest burst time may starv.
↳ Performance depends on Gantt chart
Time quantum.
↳

Ready Queue



Running Queue



Process Control Block

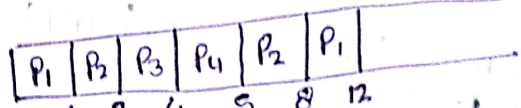
Context Switching. → Save Running Process Context / PCB

Priority Scheduling. (Preemitive mode) Criteria - Priority mode Preemitive.

Priority	Process	AT	BT	CT	TAT	WT
10	P_1	0	5	12	12	7
20	P_2	1	4	8	7	3
30	P_3	2	2	4	2	0
40	P_4	4	1	5	1	0
				24/4	22/4	10/4

min
high Priority
max

avg.



disadv → low priority will starvation
↳ poor avg,

Note:- Aging

Numericals

① FCFS — 6 Process, 1 overhead for each process, efficiency?

Proc	AT	BT	CT
P_1	0	3	4
P_2	1	2	7
P_3	2	1	9
P_4	3	4	14
P_5	4	5	20
P_6	5	2	23

Useless time or wasted time = 6

Total time = 23

Usefull time = 6/23 * 23 − 6 = 17

Efficiency η = Usefull time / Total time

= 17/23 = 0.7391

% = 73.91 %

δ	P_1	δ	P_2	δ	P_3	δ	P_4	δ	P_5	δ	P_6

0 1 2 4 5 7 8 9 10 14 15 20 21 23

TTS — Total time spent

② FCFS → AT of P_1, P_2, P_3 is zero. & 10, 20, 30 respectively of Process.

1st 20% is I/O next 80% is CPU. Utilization of CPU % = ?

Process	AT	TTS	I/O	CPU
P_1	0	10	2	8
P_2	0	20	4	16
P_3	0	30	6	24

Waste time = 2

Utilized time = 48

Efficiency η = 48/50 = .96

% = 96 %

	P_1	P_2	P_3

0 2 10 26 50

③ RR Round Robin → P_1, P_2, P_3 arrival time zero. Total Execution time 10, 15, 20 ms.

TET

20% I/O, next 60% CPU, next 20% I/O. CPU free % = ? RRS.

Process	AT	TET	I/O	CPU	I/O
P_1	0	10	2	6	2
P_2	0	15	3	9	3
P_3	0	20	4	12	4

Waste time = 6

Utilized time = 27

Efficiency η = $\frac{27}{33} \times 100$ =

CPU free % = $\frac{6}{33} \times 100 = 18.184$

	P_1	P_2	P_3	P_1	P_2	P_3	P_3	

P_1 I/O

0 2 5 12 17 18 22 27 29 33

P_2 I/O

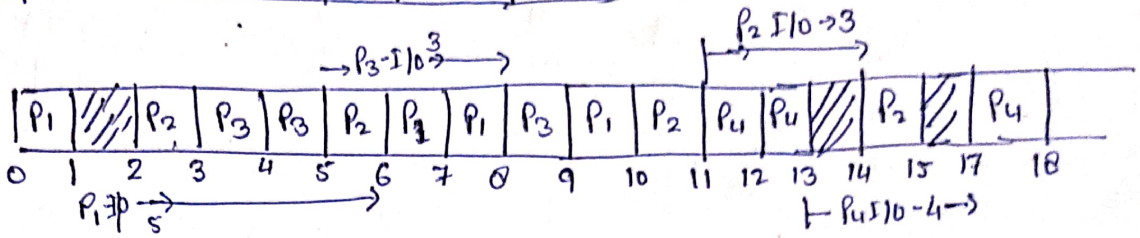
Example: for CPU & I/O Time. → mode - Premitive Criteria - Priority,

$P_3 \rightarrow P_1 \rightarrow P_2 \rightarrow P_4$
H L

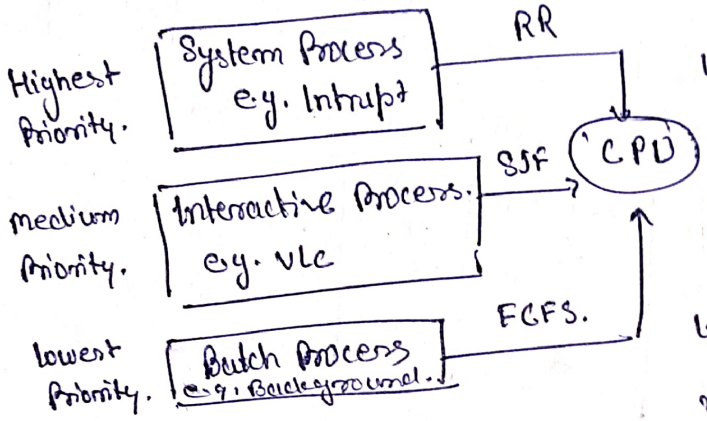
Proc	AT	Priority	CPU	I/O	CPU	CT
P_1	0	2	1	5	3	10
P_2	2	3	3	3	1	15
P_3	3	1	2	3	1	9
P_4	3	4	2	4	1	18

Ratio of CPU idealness → $\frac{\text{no. of ideal}}{\text{Total time}}$
$= 4/18$

Usage = $14/18$



Multilevel Queue Scheduling

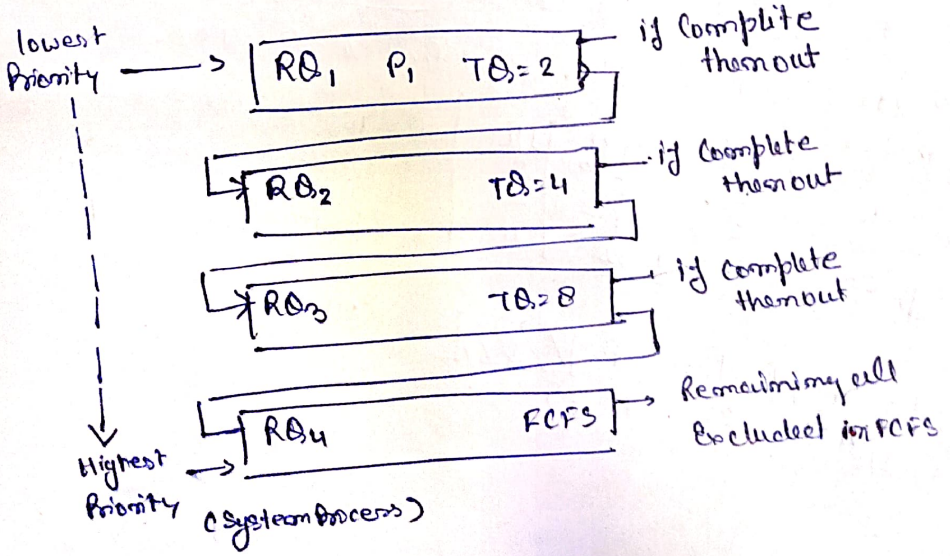


↳ Algo's can be change

↳ if. HP Process comes again & again then other Process are in waiting. that's Prob. called starvation.

↳ Overcome we can use. multilevel feedback Queue.

Multilevel feedback Queue. → Use for two Priority Process. by using feedback



KIET
GROUP OF INSTITUTIONS
Connecting Life with Learning

Department of Computer Science and Information Technology

Operating System (BCS 401)

Process States and State Transition Diagram

By: Dr. Ankur Garg, Professor

Topic(s) Covered

- Process States
- State Transition Diagram
- Different Scheduling Queues used in State Transition Diagram
- Working of Schedulers in State Transition Diagram

Process State

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process.

Each process may be in one of the following states:

New: The process is being created.

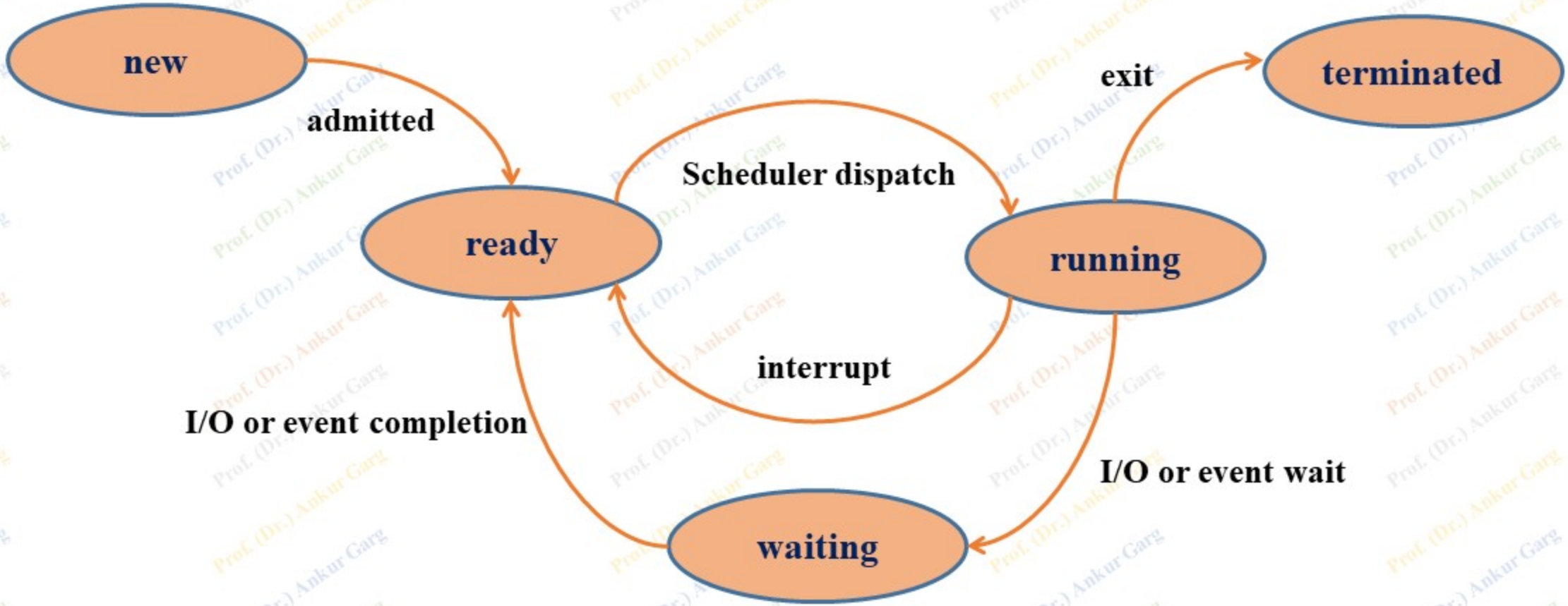
Ready: The process is waiting to be assigned to a processor.

Running: Instructions are being executed.

Waiting: The process is waiting for some event to occur (such as an I/O completion or reception of a signal).

Terminated: The process has finished execution

State Transition Diagram



- A process migrates between the various **scheduling queues** throughout its lifetime.

- The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate **scheduler**.

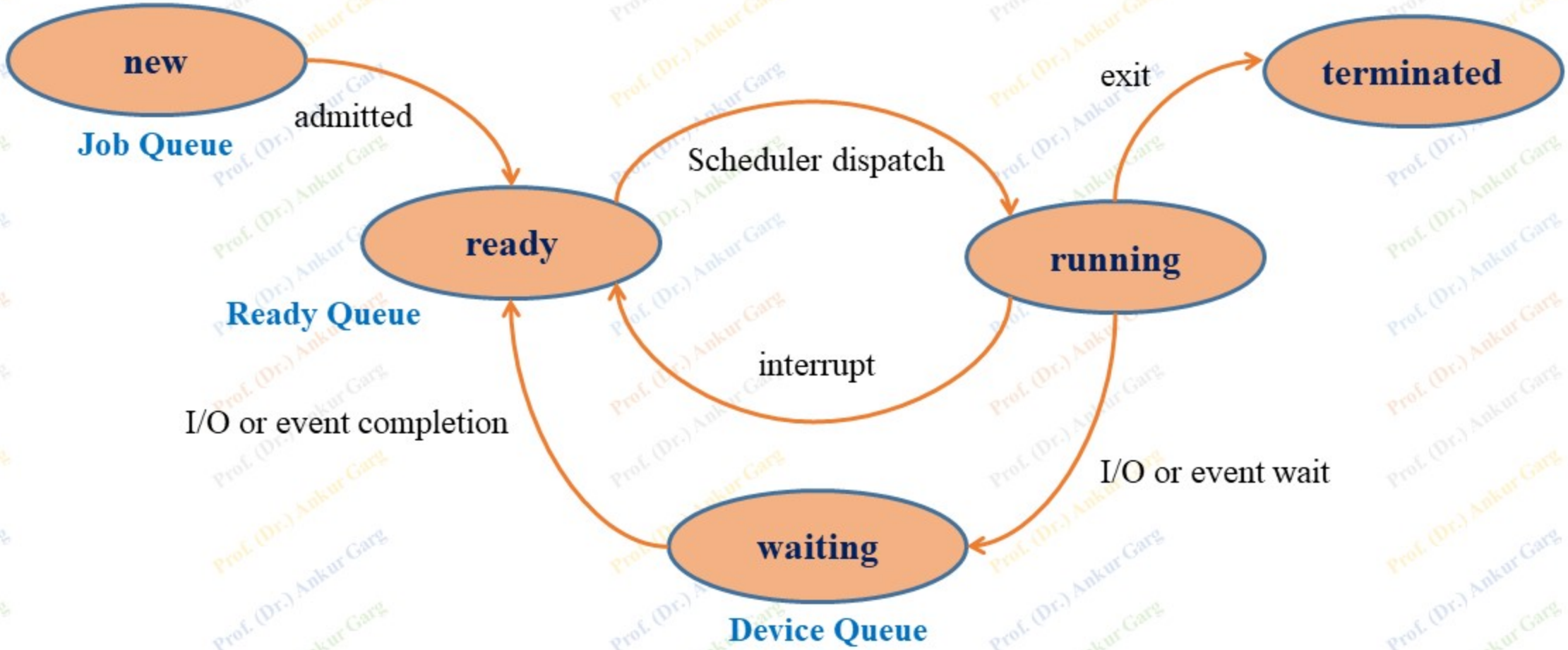
Different Scheduling Queues used in State Transition Diagram

i) **Job queues:-** As process enter the system, they are all put in to the queue called job queue.

ii) **Ready queue:-** The process that are residing in main memory and are ready and waiting to execute are kept in a list called ready queue. This queue is generally stored as a linked list. A ready queue header will contain pointers to the first and last PCB's in the list. Each PCB has a pointer field that points to the next process in the ready queue.

iii) **Device queue:-** Some times many processes need to wait for an I/O device such as disk, printer or tape etc… the list of processes waiting for a particular I/O device has its own device queue. Each device has its own device queue.

Scheduling Queues



Working of Scheduler in State Transition Diagram

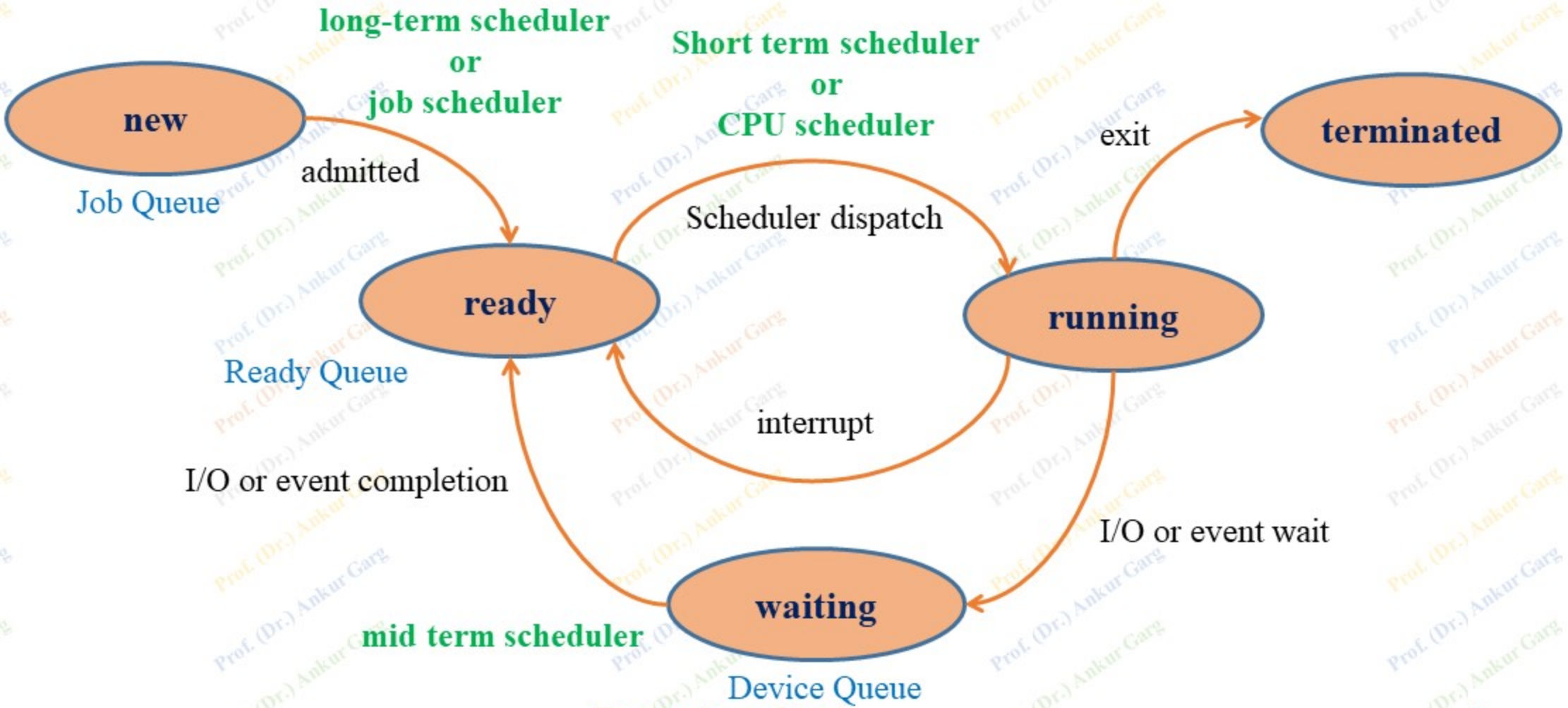
- **The long-term scheduler, or job scheduler,** selects processes from a pool (i.e. processes are spooled to a mass-storage device typically a disk) and loads them into memory for execution.

- **The short-term scheduler, or CPU scheduler,** selects from among the processes that are ready to execute, and allocates the CPU to one of them.

- **The medium-term scheduler** removes processes from memory (and from active contention for the CPU), and thus reduces the degree of multiprogramming. At some later time, the process can be reintroduced into memory and its execution can be continued where it left off. This scheme is called swapping.

The process is swapped out, and is later swapped in, by the medium-term scheduler.

Scheduler(s)


long-term scheduler
or
job scheduler
Short term scheduler
or
CPU scheduler
new
admitted
exit
terminated
Job Queue
Scheduler dispatch
ready
running
Ready Queue
interrupt
I/O or event completion
I/O or event wait
waiting
mid term scheduler
Device Queue

Thank You