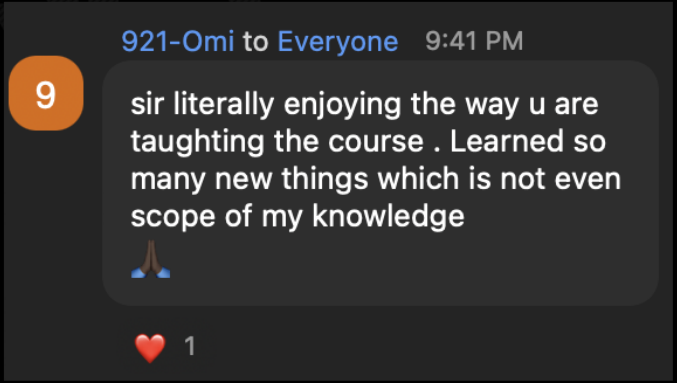
Lecture 8

Topic: Process States

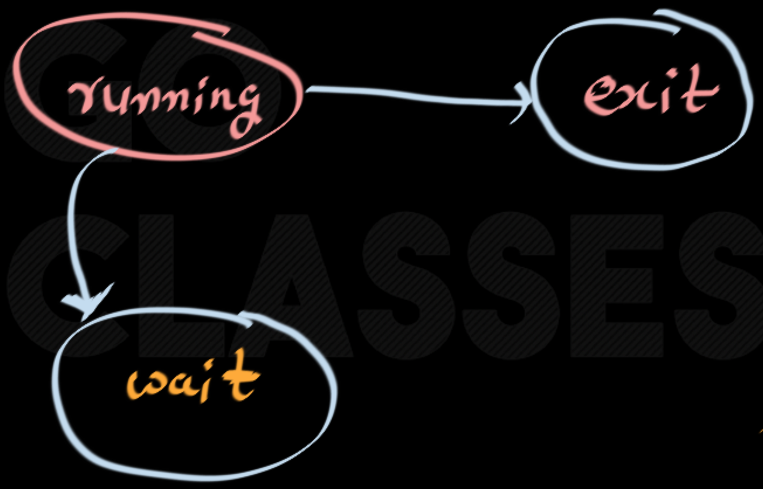
Topic2: Types of Schedulers

Topic 3: Scheduling Criteria

Topic 4: Scheduling Algorithms (FCFS)



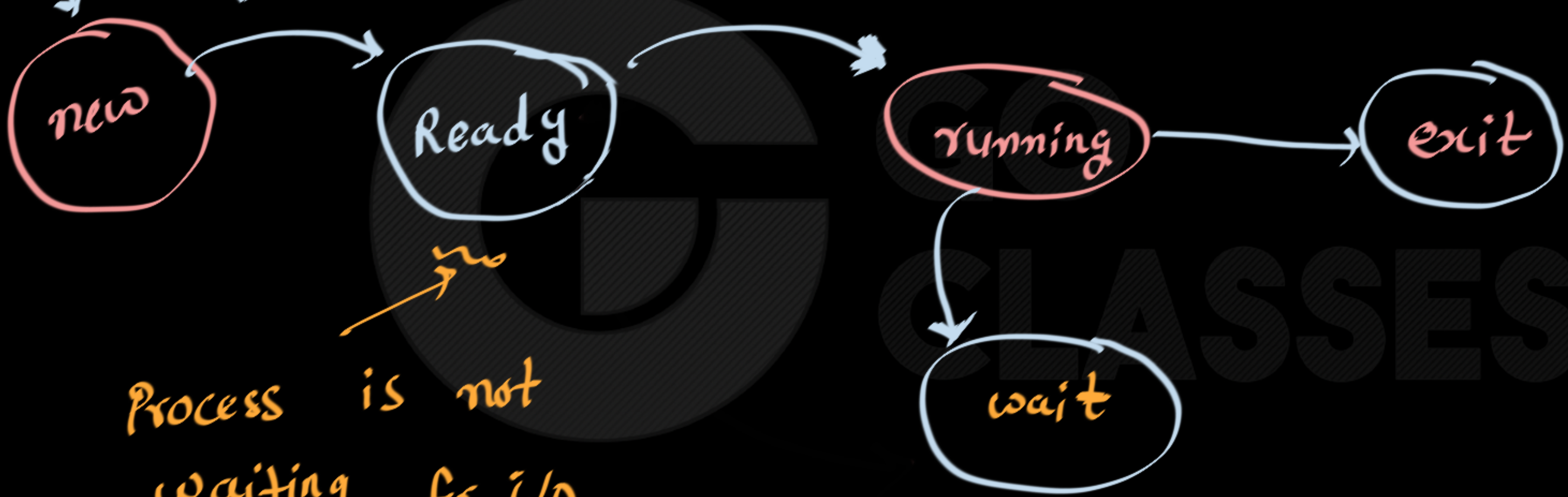
Process States



running
exit
wait

./a.out fork_exec

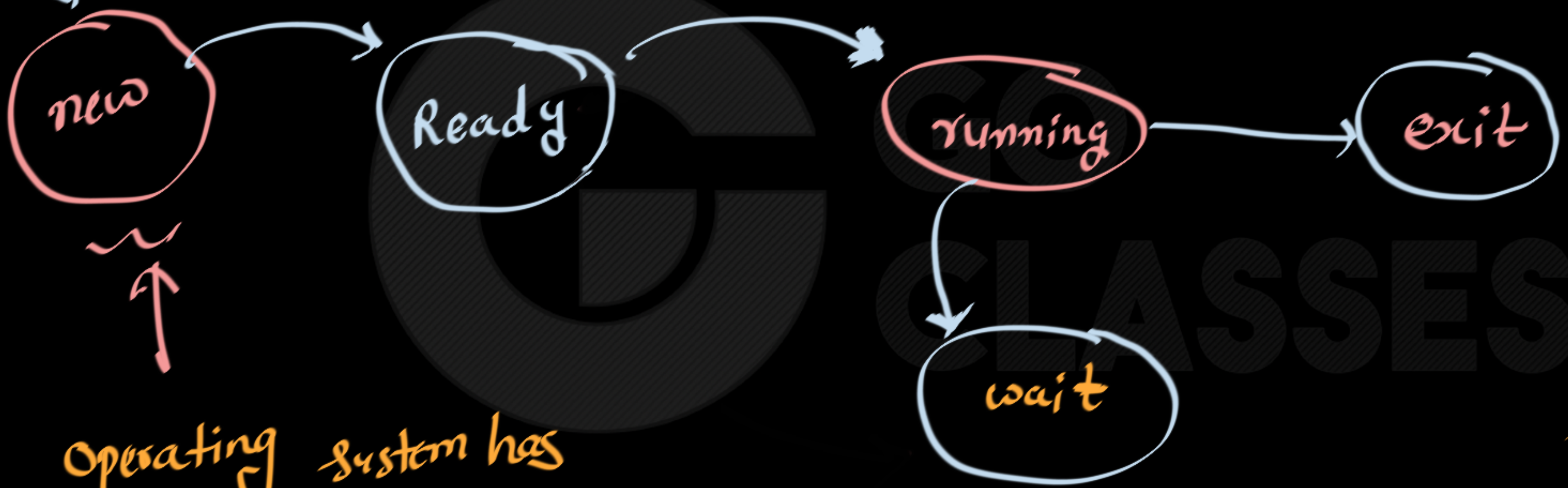
Process is idle and sitting in mm
(it wants to Run)



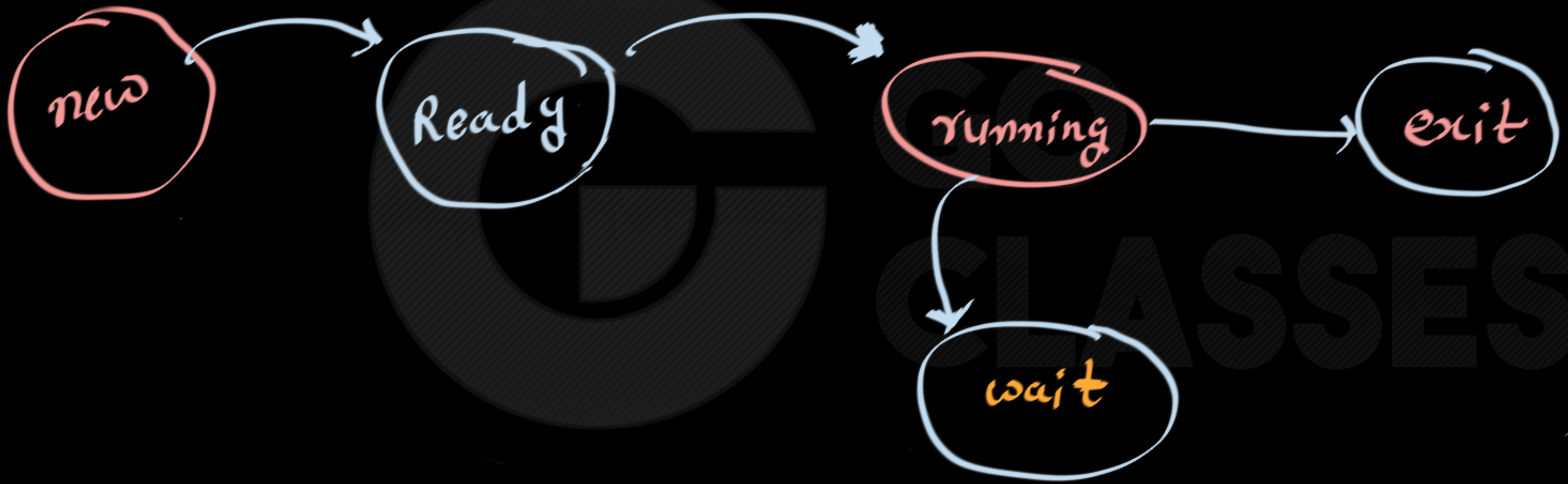
Process is not
waiting for i/o
neither running on
CPU (and in mm)

./a.out   fork_exec

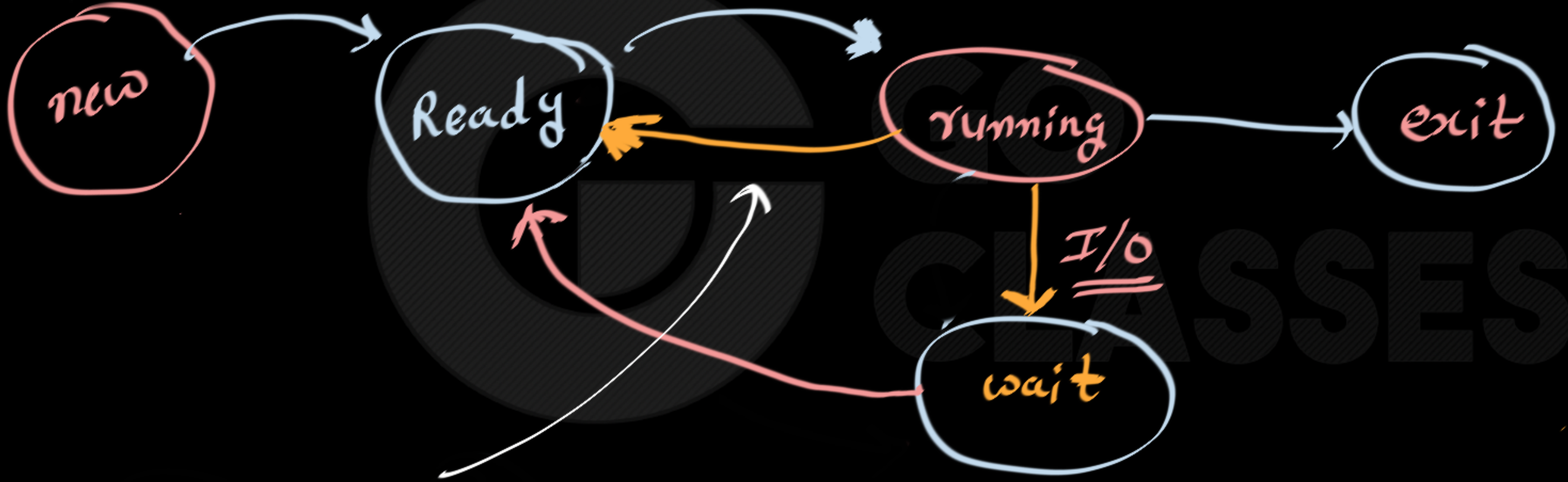
Process is idle and sitting in mm
(it wants to Run)



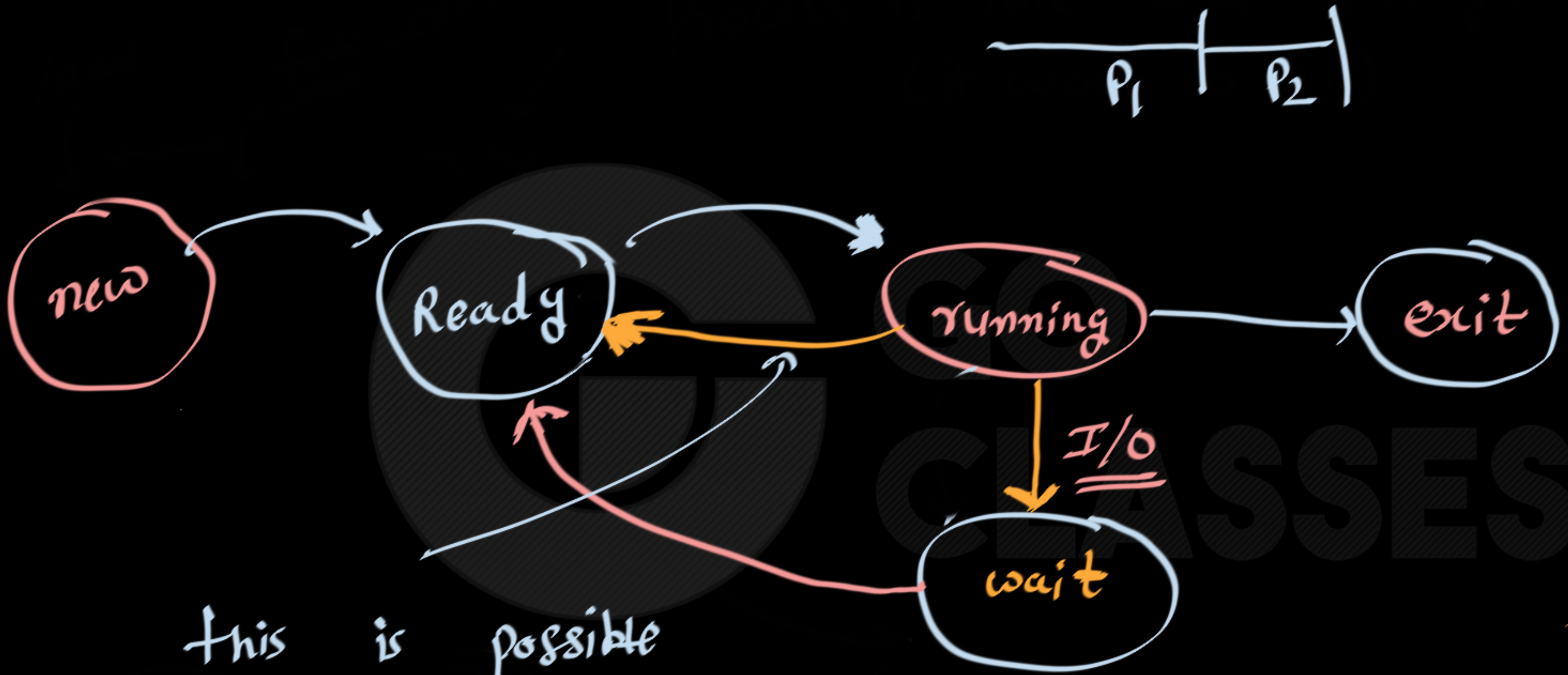
Operating system has got a request to create a NEW process



This is Just ONE possible set of states, if os desiner need he/she can have more states

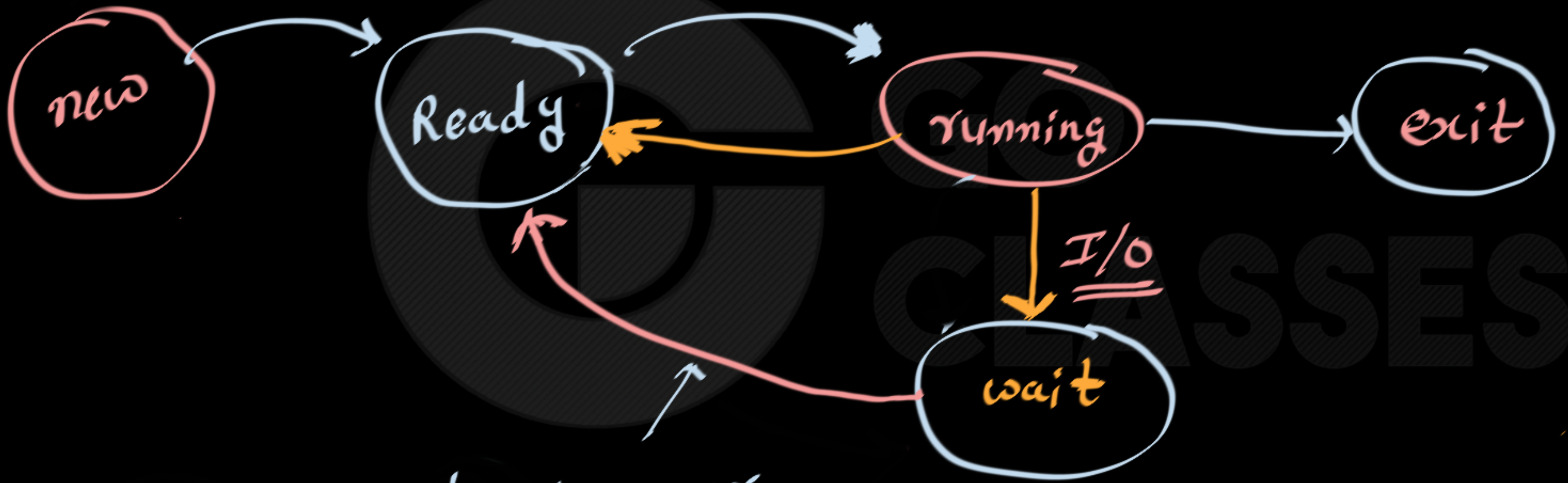


because of some reason
OS can decide to put running process in the ready queue.



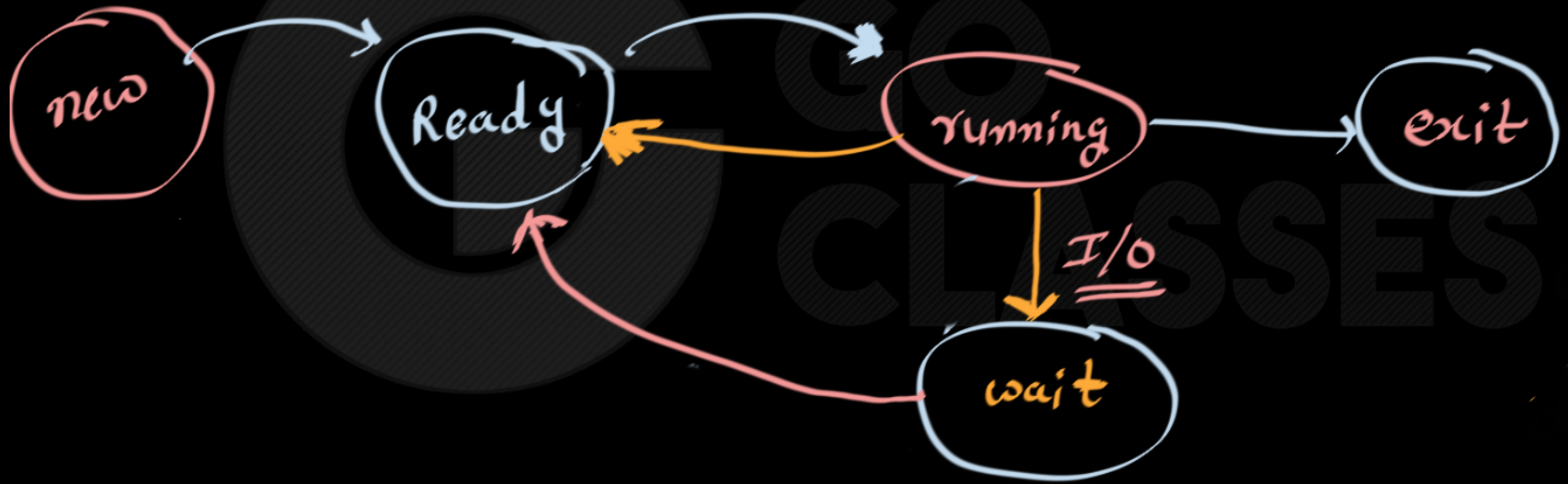
this is possible

only when we (os) are using preemptive scheduling algorithm



I/O is over
and now i want CPU.

5 - state diagram.

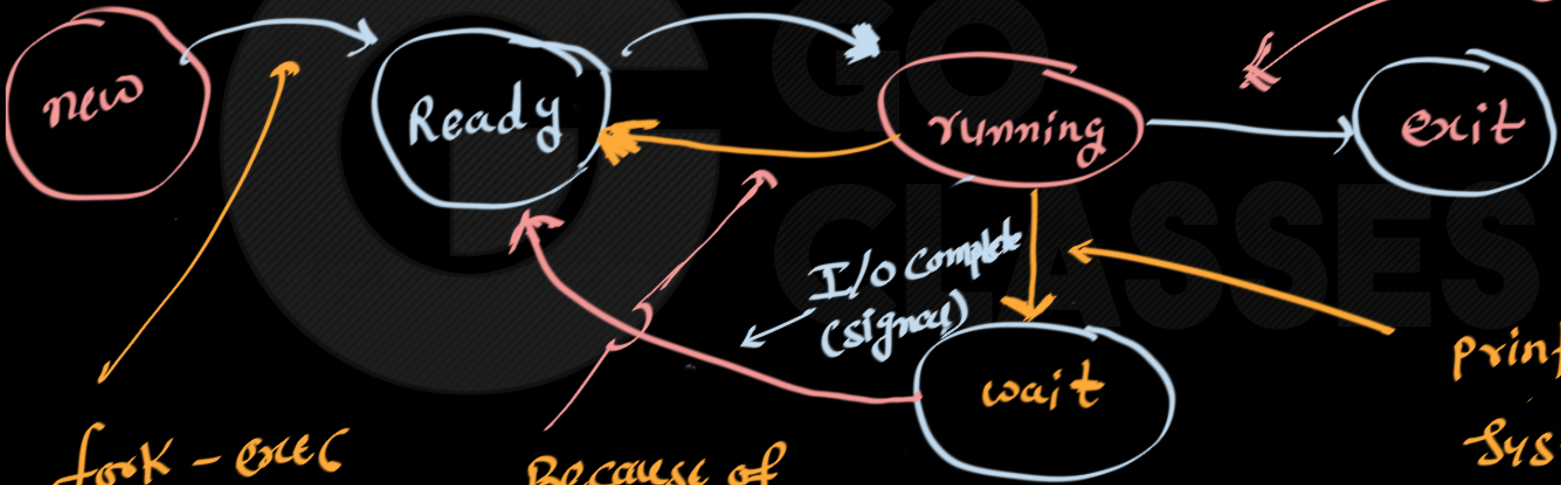


5 - state diagram.

event driven transitions

Scheduler has decided

exit system call



printf
System call which require I/O

fork - exec

Because of some scheduling criteria (Hardware timer interrupt)

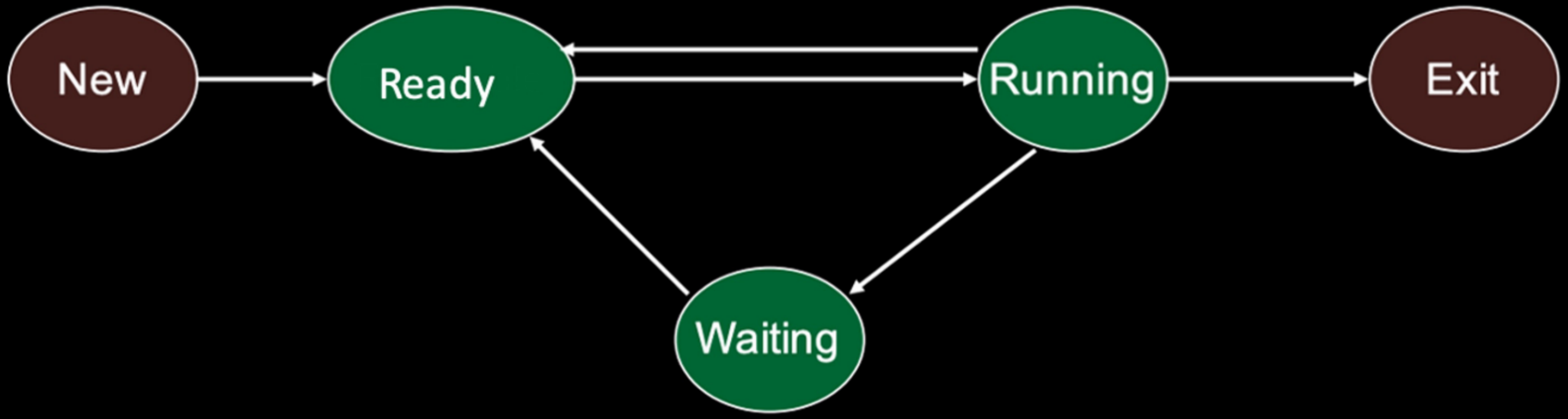
How do we create a process?

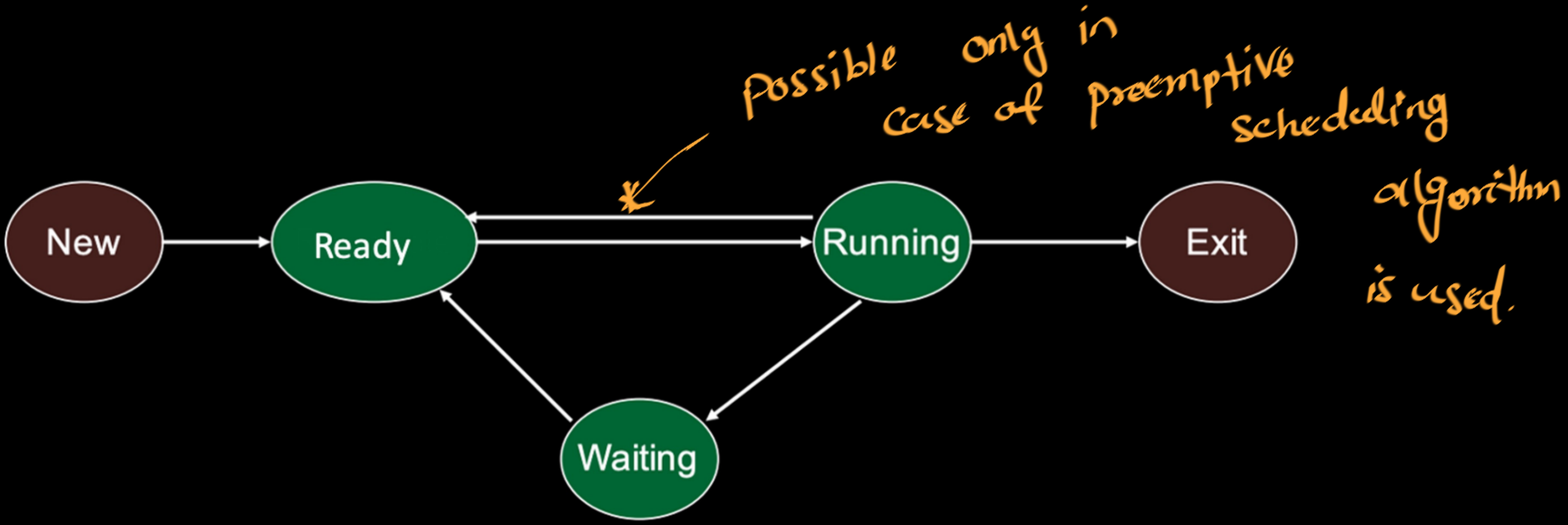
./a.out } command ← you want to run a process.

fork
exec

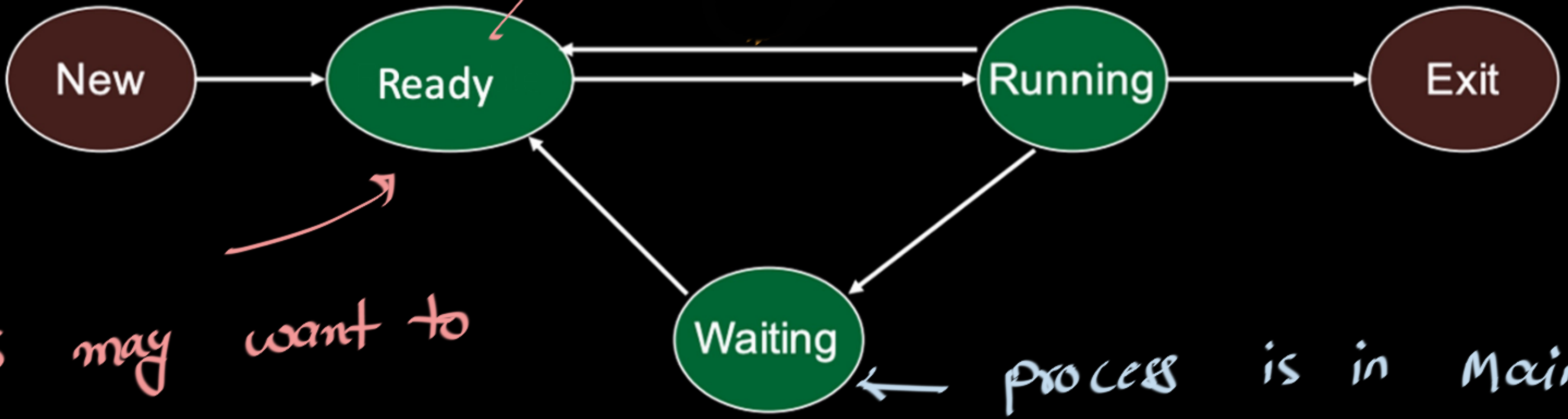
./a.out

fork
exec


New
Ready
Running
Exit
Waiting

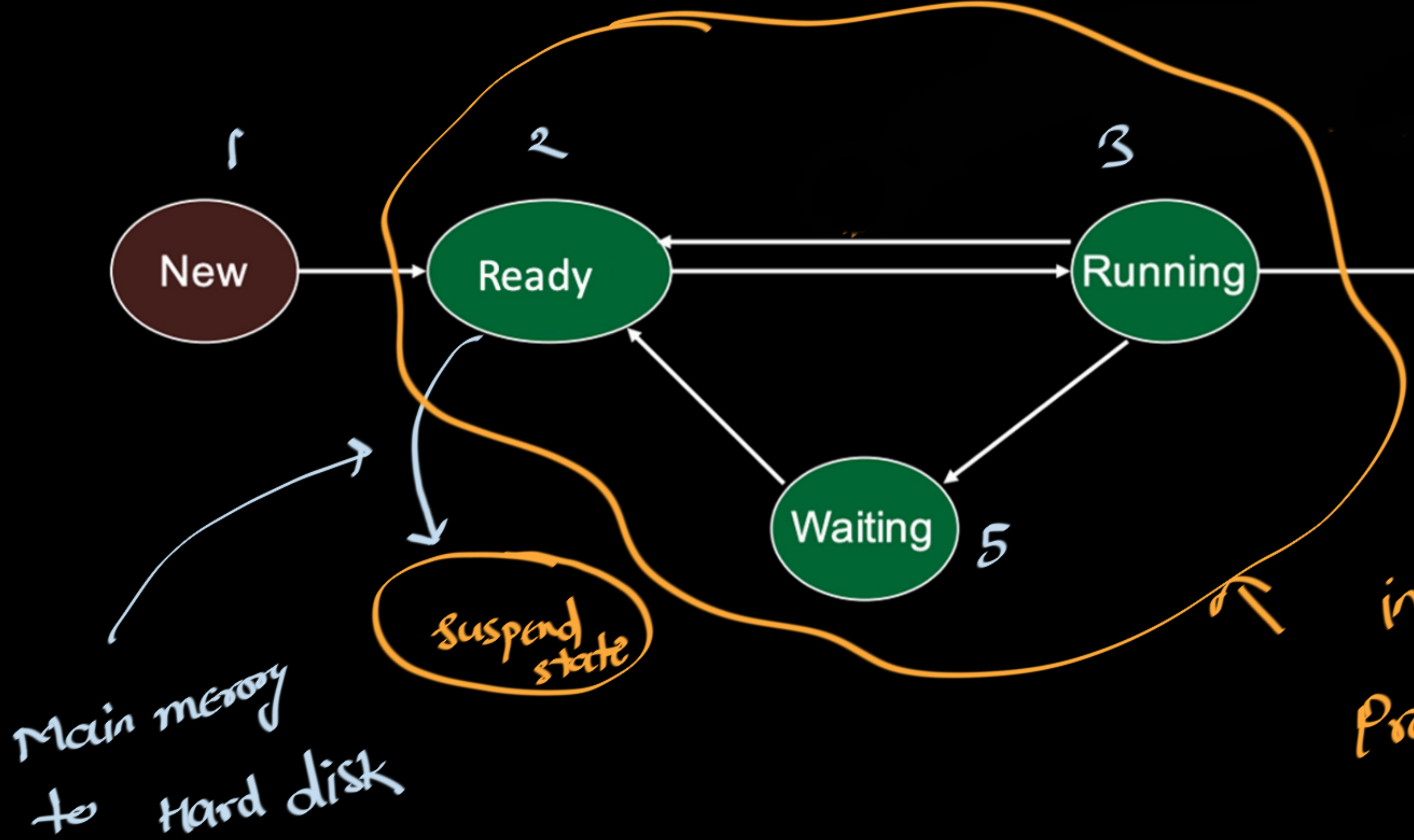

Possible only in case of preemptive scheduling algorithm is used.
*
New
Ready
Running
Exit
Waiting

process is in MM

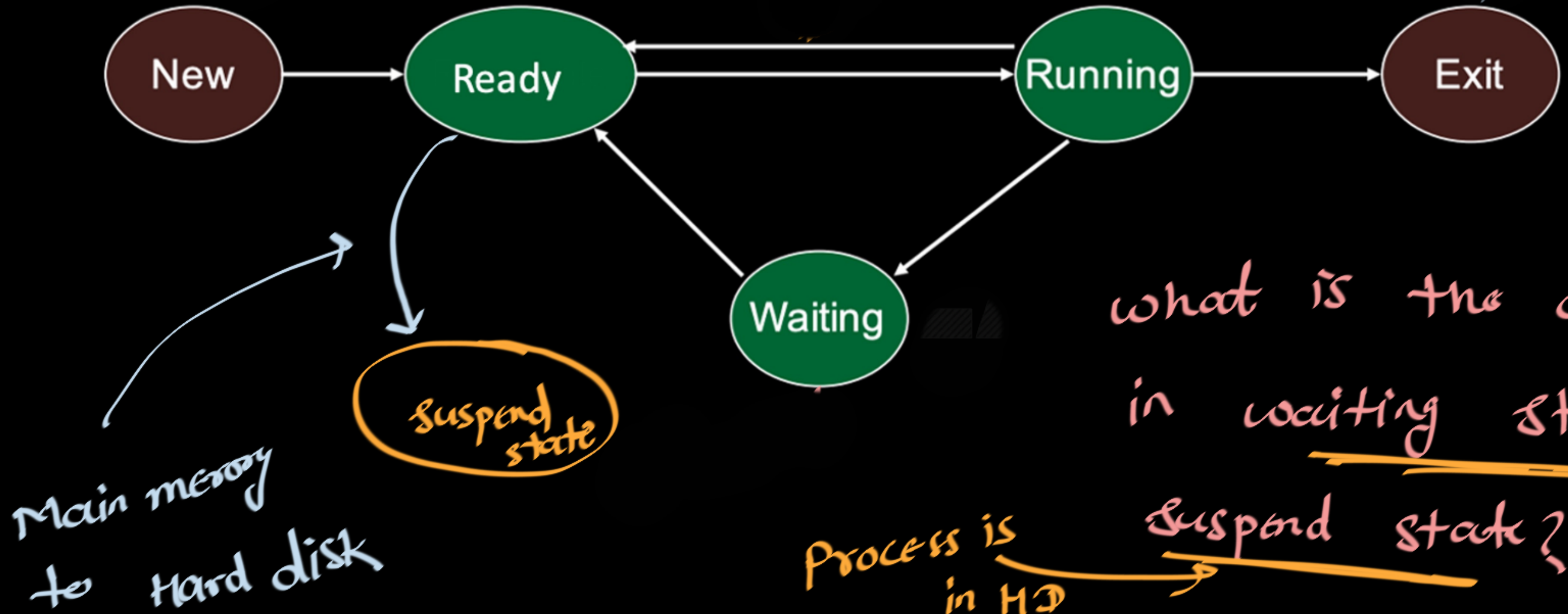


process is in Main memory

OS may want to put some processes from ready state to Hard disk



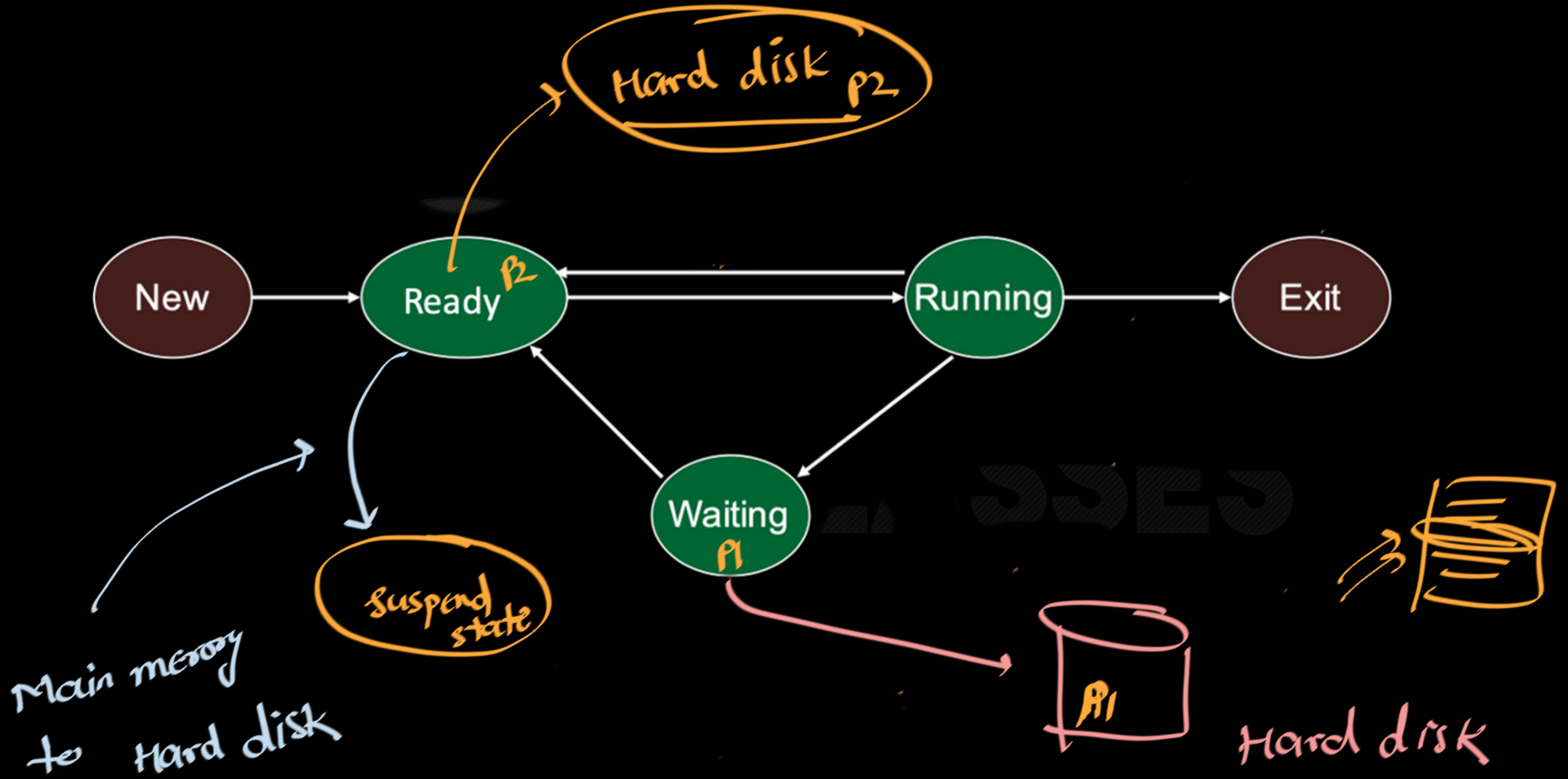
in these 3 states Process is in MM



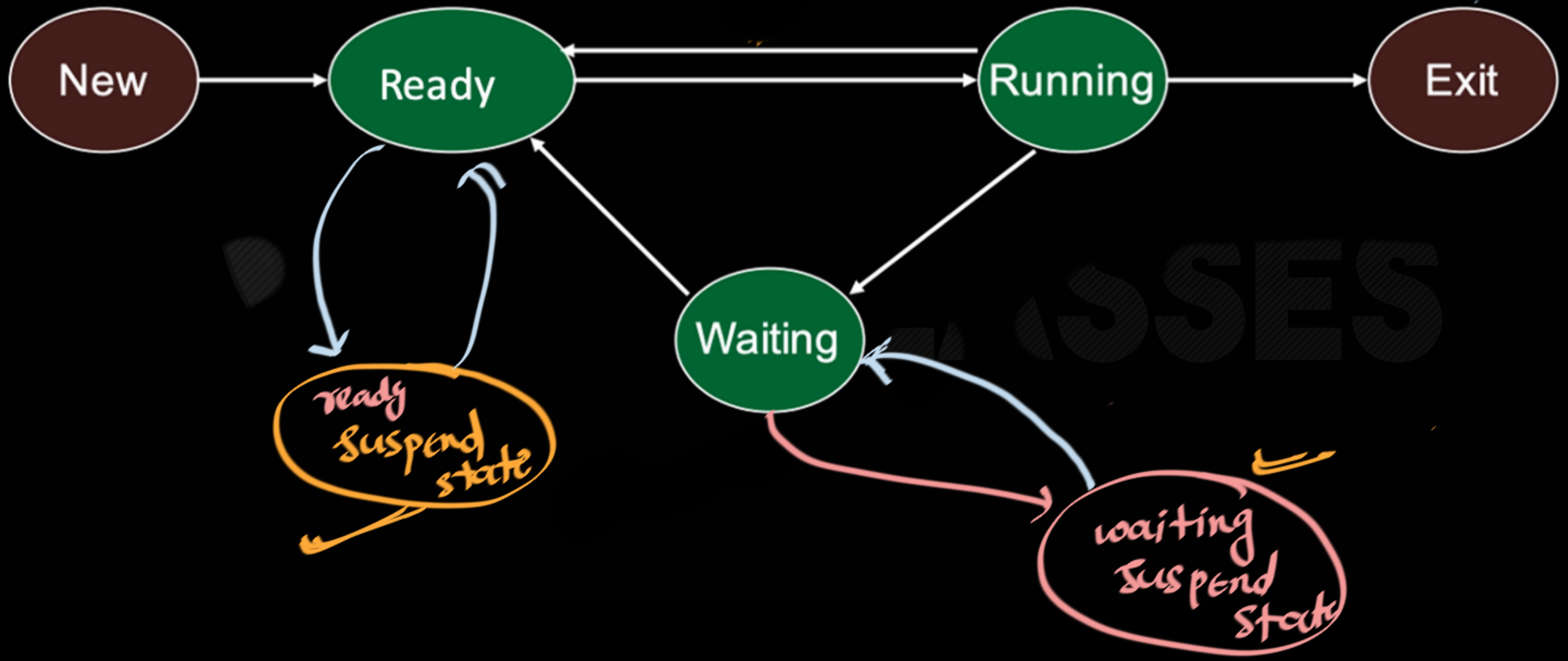
what is the difference in waiting state and suspend state?

Process is in HD

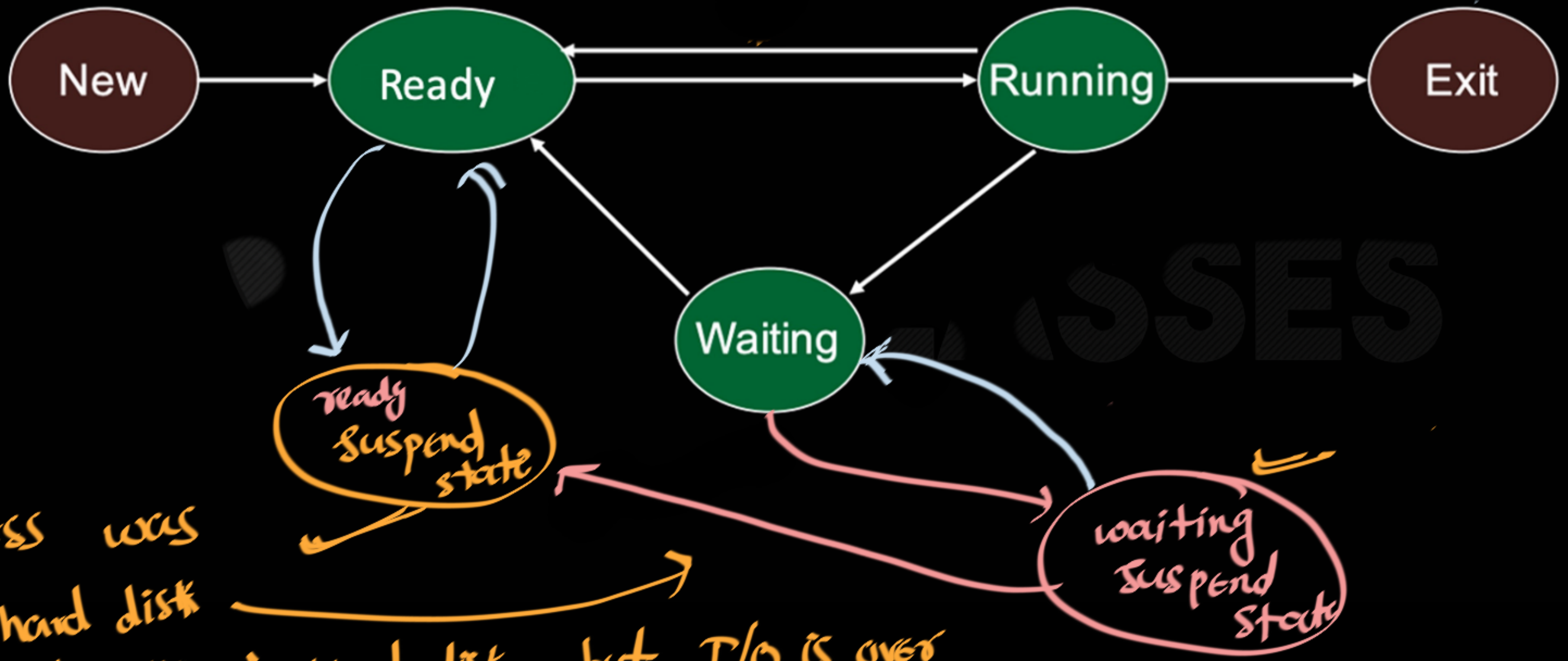
process is in mm


Hard disk P2
New
Ready P2
Running
Exit
Waiting P1
suspend state
Main memory to Hard disk
P1
Hard disk

7 - state diagram



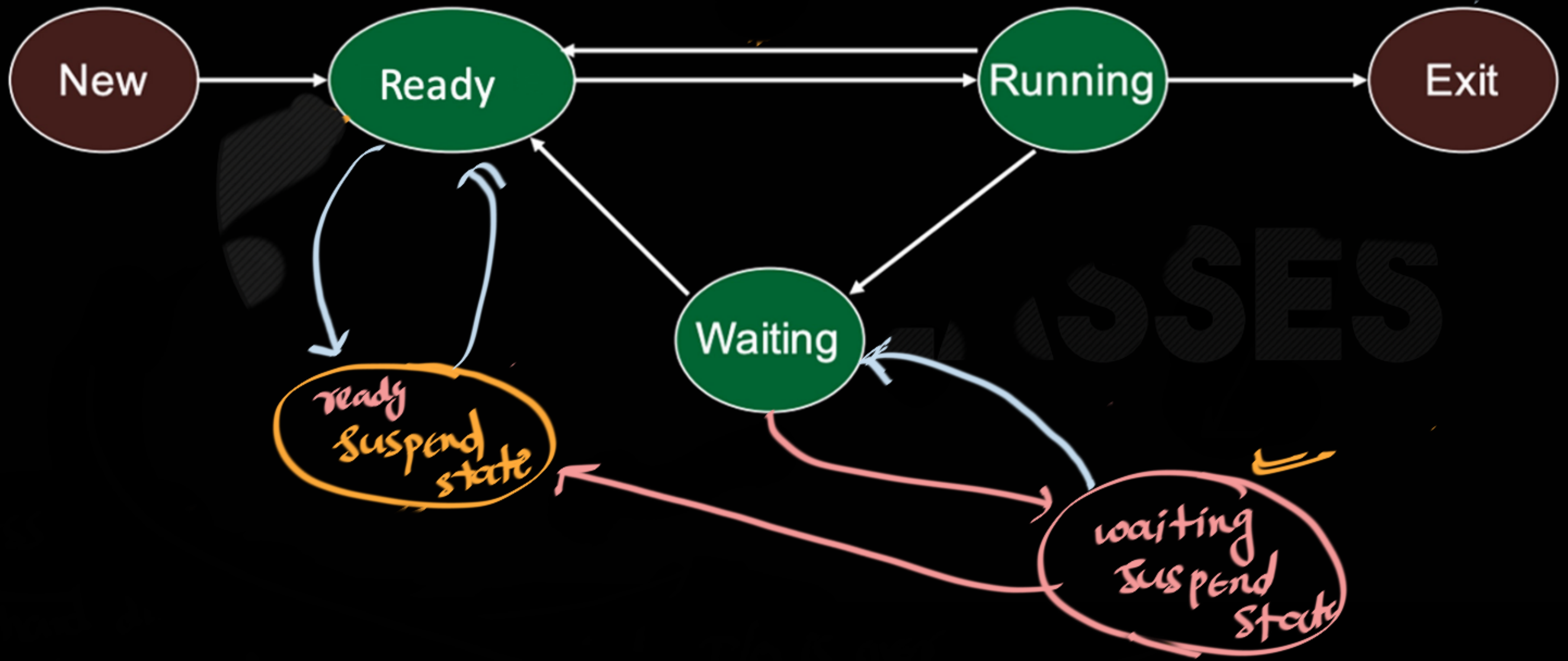
7 - state diagram



Process was in hard disk and still in Hard disk but I/O is over

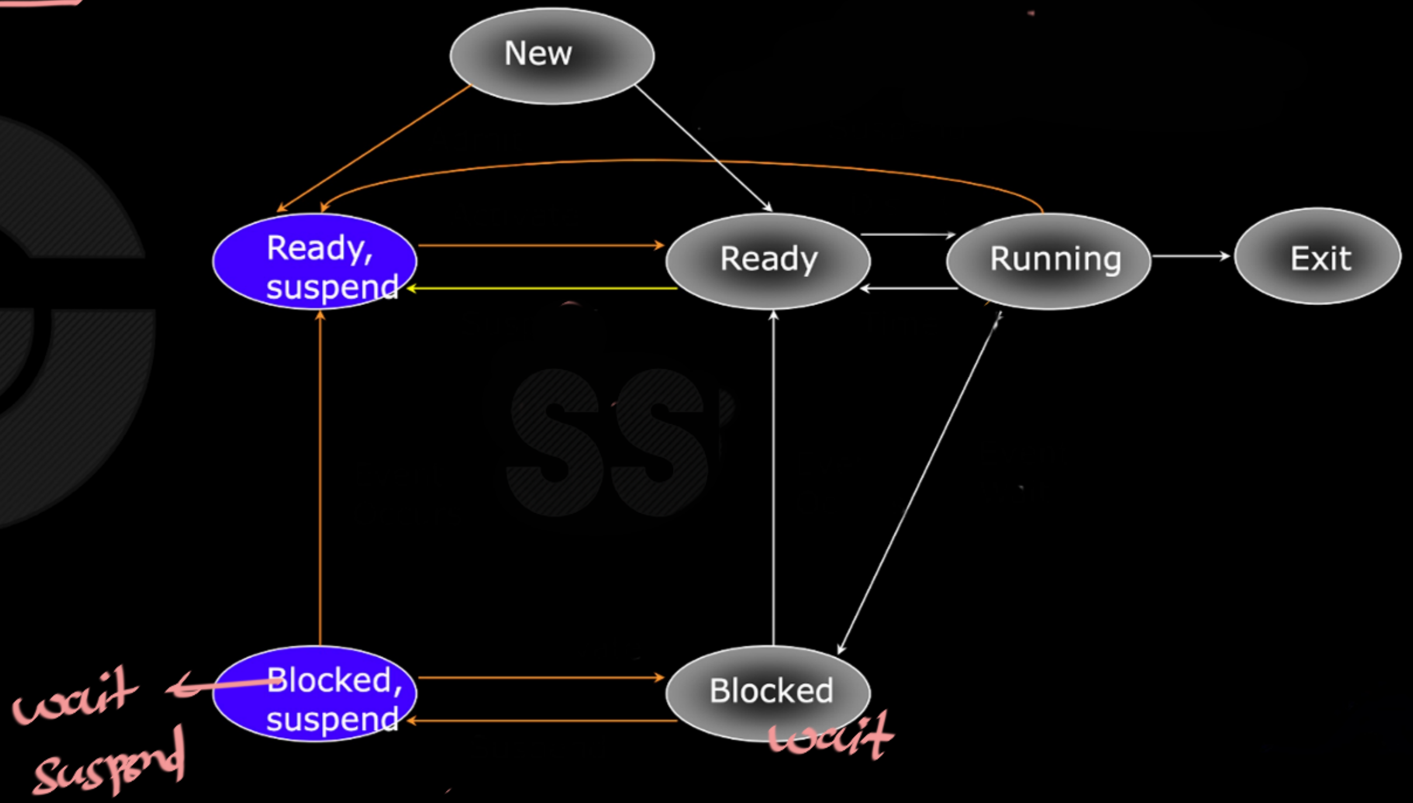
GO CLASSES

7 - state diagram



Optional: 7 state process diagram

Process State Diagram with 2 suspended States



One thing about one COA topic

as per von neuman arch. we need any process to be in MM if we want cpu to execute it.

Otherwise (if only I/O) then we can put it in MM or secondary MM

New Topic :

Types of Schedulers

Types of Schedulers

Operating systems may feature up to 3 distinct types of schedulers:

- Long-term scheduler — Task 1 } ← very less frequent set of tasks
- Mid-term scheduler — Task 2 } ← Not that frequent tasks
- Short-term scheduler — Task 3 ← too frequent tasks

Types of Schedulers

Operating systems may feature up to 3 distinct types of schedulers:

- Long-term scheduler
- Mid-term scheduler
- Short-term scheduler

The names suggest the relative frequency with which these functions are performed.

Types of Schedulers

Operating systems may feature up to 3 distinct types of schedulers:

- Long-term scheduler new → Ready
- Mid-term scheduler Ready → Ready Suspend
  wait/block → Block Suspend
- Short-term scheduler
  Ready → Running

Ready

CPU

Long-term scheduler

Ready state is full

new → Ready

8GB Hard disk

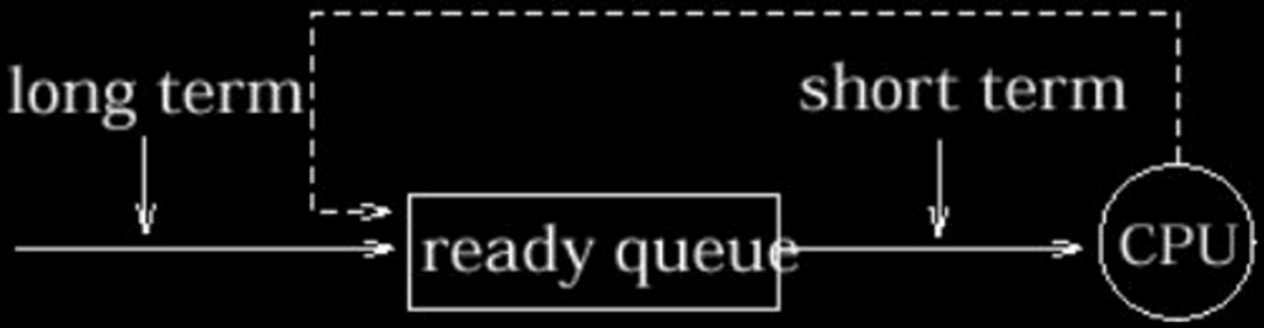
if process is submitted to OS then

→ where to put in ready queue

→ to put or not to put in ready queue

these kind of decisions are taken by long term scheduler.

Long-term scheduler



Typically for a desktop computer, there is no long-term scheduler as such, and processes are admitted to the system automatically.

However this type of scheduling is very important for a real-time operating system, as the system's ability to meet process deadlines may be compromised by the slowdowns and contention resulting from the admission of more processes than the system can safely handle.

RR

RTOS

https://lass.cs.umass.edu/~shenoy/courses/fall08/lectures/Lec07_notes.pdf long term schedulers are invoked with very less frequency (seconds / minutes / hours - -)

we call it "long term" because we invoke it after a long time (since we invoke it occassionaly hence it is OK for us even take more time

Mid-term Scheduler

The mid-term scheduler, present in all systems with virtual memory, temporarily removes processes from main memory and places them on secondary memory (such as a disk drive) or vice versa.

main memory but not using CPU

Hard disk

midterm scheduler

Short-term Scheduler

Ready $\xrightarrow{}$ Running
Short term

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the **short-term scheduler**, or CPU scheduler. The scheduler selects a process from the processes in memory that are ready to execute and allocates the CPU to that process.

Note that the ready queue is not necessarily a first-in, first-out (FIFO) queue. As we shall see when we consider the various scheduling algorithms, a ready queue can be implemented as a FIFO queue, a priority queue, a tree, or simply an unordered linked list. Conceptually, however, all the processes in the ready queue are lined up waiting for a chance to run on the CPU. The records in the queues are generally process control blocks (PCBs) of the processes.

← Galvin

→ Optional read.

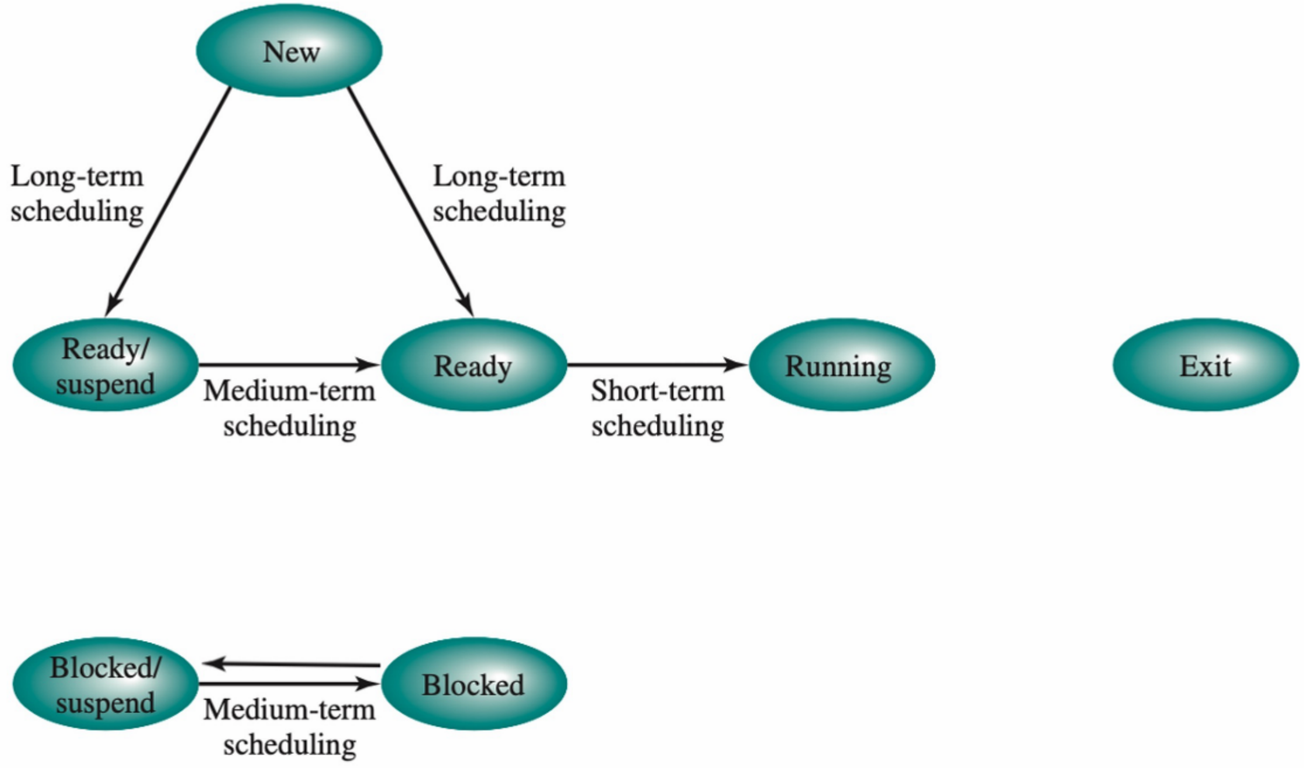


Figure 9.1 **Scheduling and Process State Transitions**

Source: Tanenbaum

- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) $\Rightarrow$ (must be fast)
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) $\Rightarrow$ (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**

New topic :

Scheduling Criteria