Process and its States

Process

- A program in execution is called process.
- A process is more than the set of instructions, besides it contains memory area, local, global variable, CPU registers, Program Counters.

Process Concept

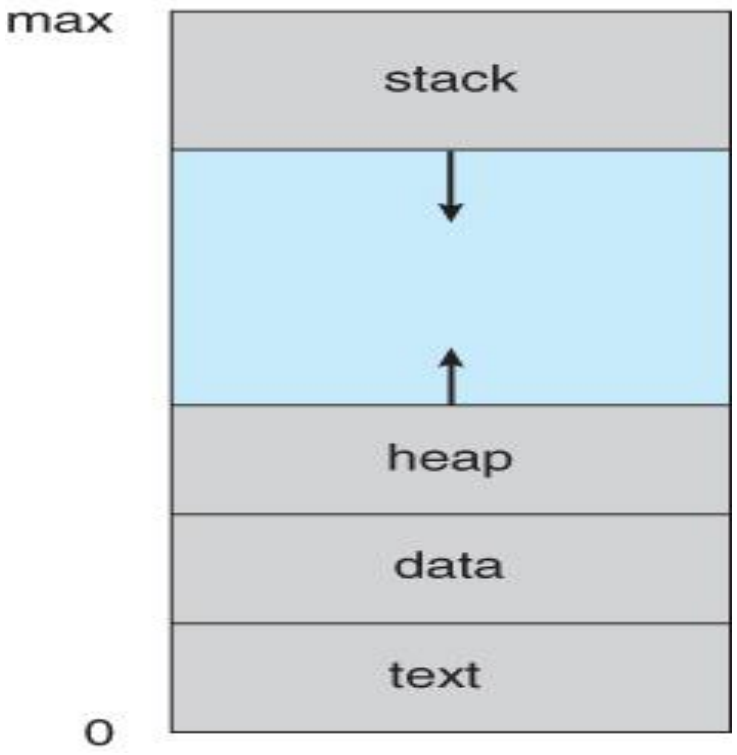
- Program is **passive** entity stored on disk (**executable file**); process is **active**
  - Program becomes process when an executable file is loaded into memory
- Execution of program started via GUI mouse clicks, command line entry of its name, etc.
- One program can be several processes
  - Consider multiple users executing the same program
  - A user running multiple copies of web browser

Process Concept

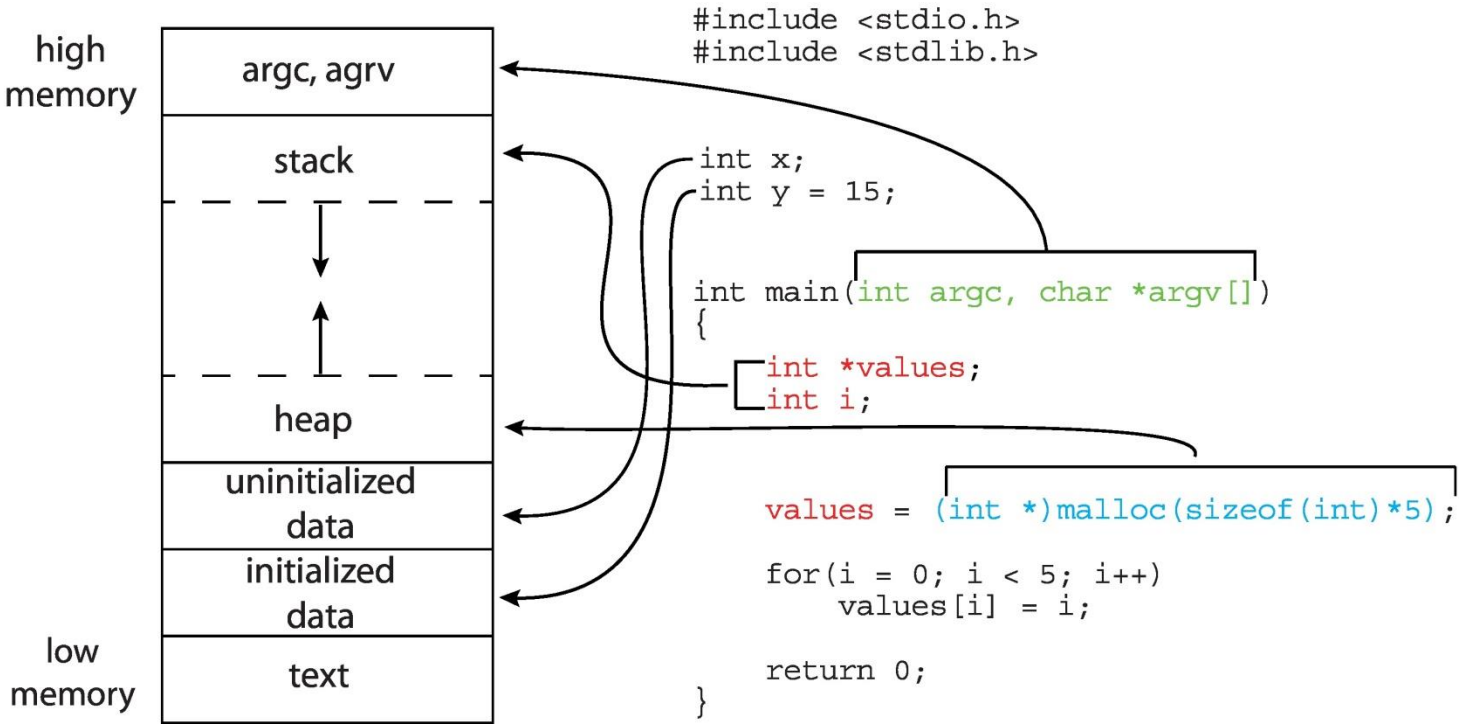
- Multiple parts
  - The program code, also called **text section**
  - Current activity including **program counter**, processor registers
  - **Stack** containing temporary data
    - Function parameters, return addresses, local variables
  - **Data section** containing static and global variables
  - **Heap** containing memory dynamically allocated during run time

Process in Memory

- When a program is loaded into the memory and it becomes a process, it can be divided into four sections — stack, heap, text and data.



Memory Layout of a C Program



P-1

- which section of a program does not contain in the memory?

- Heap
- Data
- Stack
- Program counter

Types of Processes

- CPU Bound Process: If the process is intensive in terms of CPU operations then it is called CPU bound process.
- For. e.g. Multiply two matrices

- I/O Bound Process: If the process is intensive in terms of I/O operations then it is called IO bound process.
- For e.g. Word processor, counting number of lines in a file

Concept of Multiprogramming

- Pre-emption – Process is forcefully removed from CPU. Pre-emption is also called as time sharing or multitasking.

- Non pre-emption – Processes are not removed until they complete the execution. Multiprogramming is an example of Non-pre-emptive scheduling algorithm.

Process Control Block (PCB)

- When the process is created by the operating system it creates a data structure to store the information of that process.
- Each process is represented in the OS by a PROCESS CONTROL BLOCK (PCB)- also called Task control Block.

Process Id

Program counter

CPU registers

Process state

I/O status information

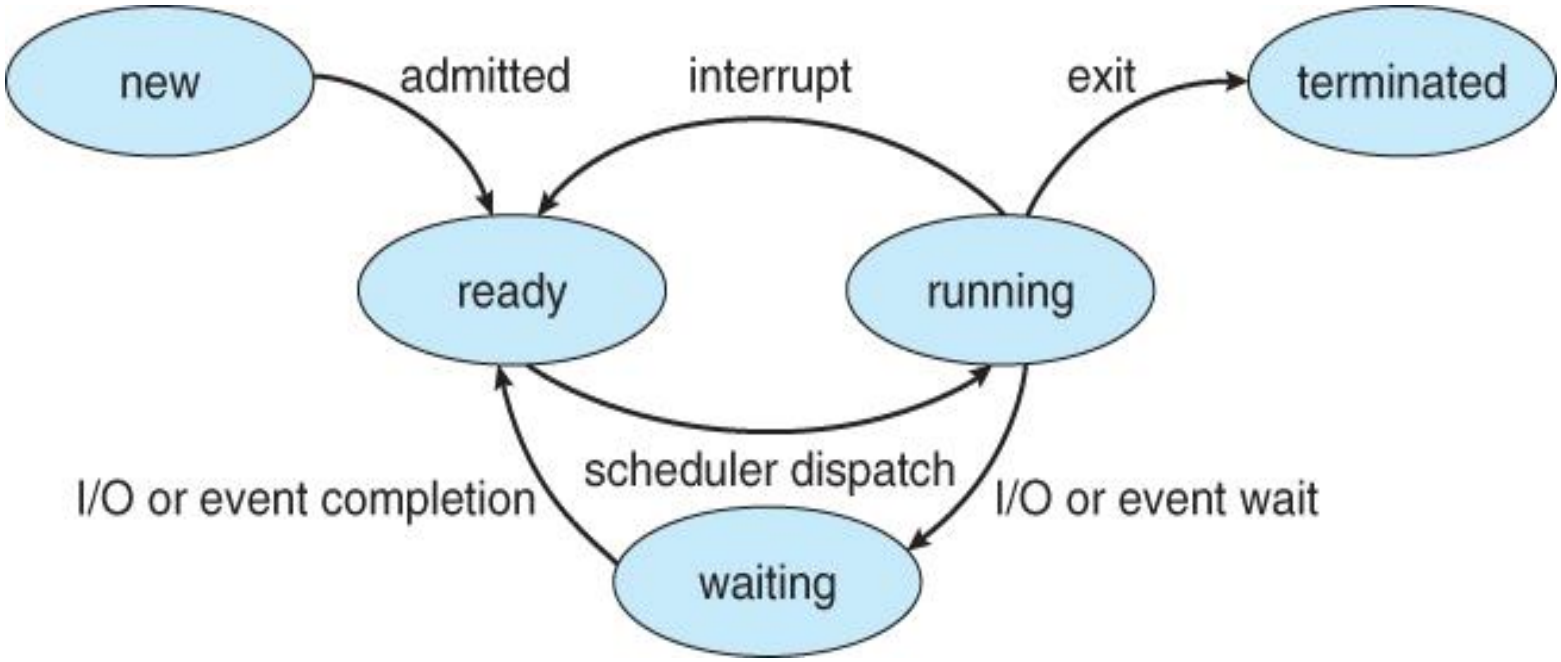
Accounting information

CPU scheduling information

Memory Management information

- PCBs are stored in specially reserved memory for the operating system known as kernel space.
- The Random Access Memory (RAM) can be logically divided into two distinct regions namely –
  - the kernel space and the user space.
- kernel space is the core of the operating system. It normally has full access to all memory and machine hardware and it cant be accessed by the user.

Process State Diagram (5-States)



Awake (Process-name) : New -> Ready
Dispatch (Process-name): ready -> running
Timerrunout (Process-name): running -> ready
Block(Process-name): running -> blocked
wakeup(Process-name): blocked-> ready

P-3

- A Process Control Block(PCB) does not contain which of the following :

- A. Program counter
- B. List of open files
- C. Process State
- D. I/O status information
- E. bootstrap program

P-2

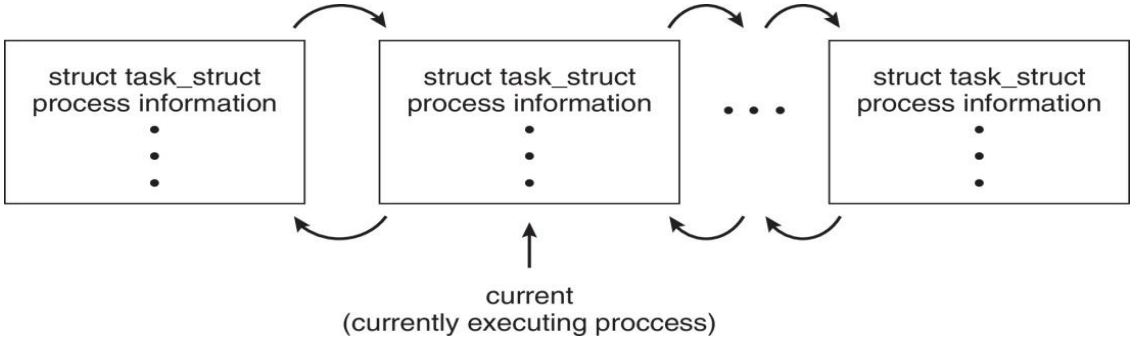
- The address of the next instruction to be executed by the current process is provided by the:

- A. CPU registers
- B. program counter
- C. process stack
- D. pipe

Process Representation in Linux

Represented by the C structure `task_struct`

```
pid t_pid;                    /* process identifier */
long state;                   /* state of the process */
unsigned int time_slice       /* scheduling information */
struct task_struct *parent;/* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files;/* list of open files */
struct mm_struct *mm;         /* address space of this process */
```

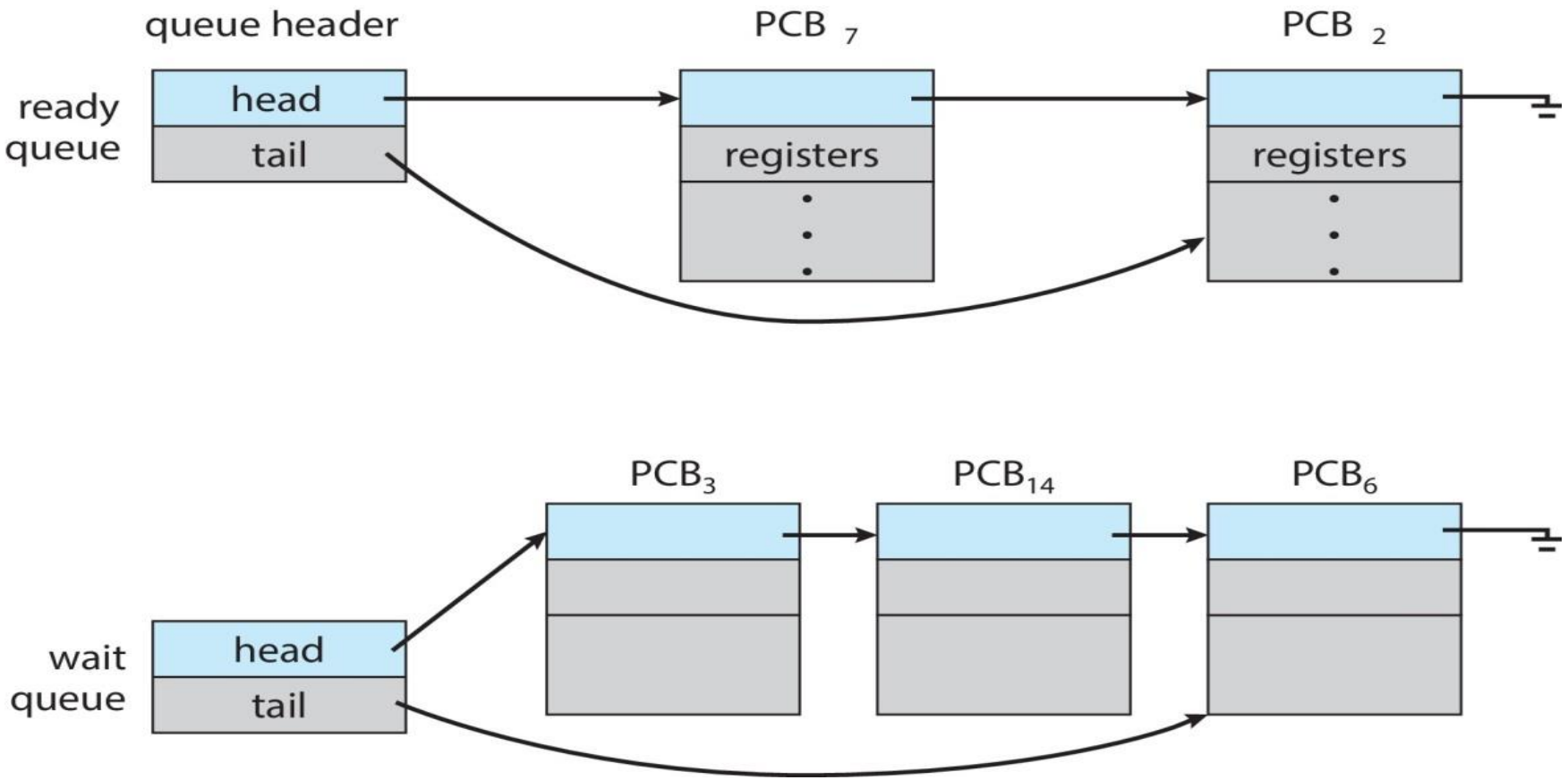


Process Scheduling

- The act of determining which process is in the **ready** state, and should be moved to the **running** state is known as **Process Scheduling**.
- Maintains **scheduling queues** of processes
  - **Job Queue**

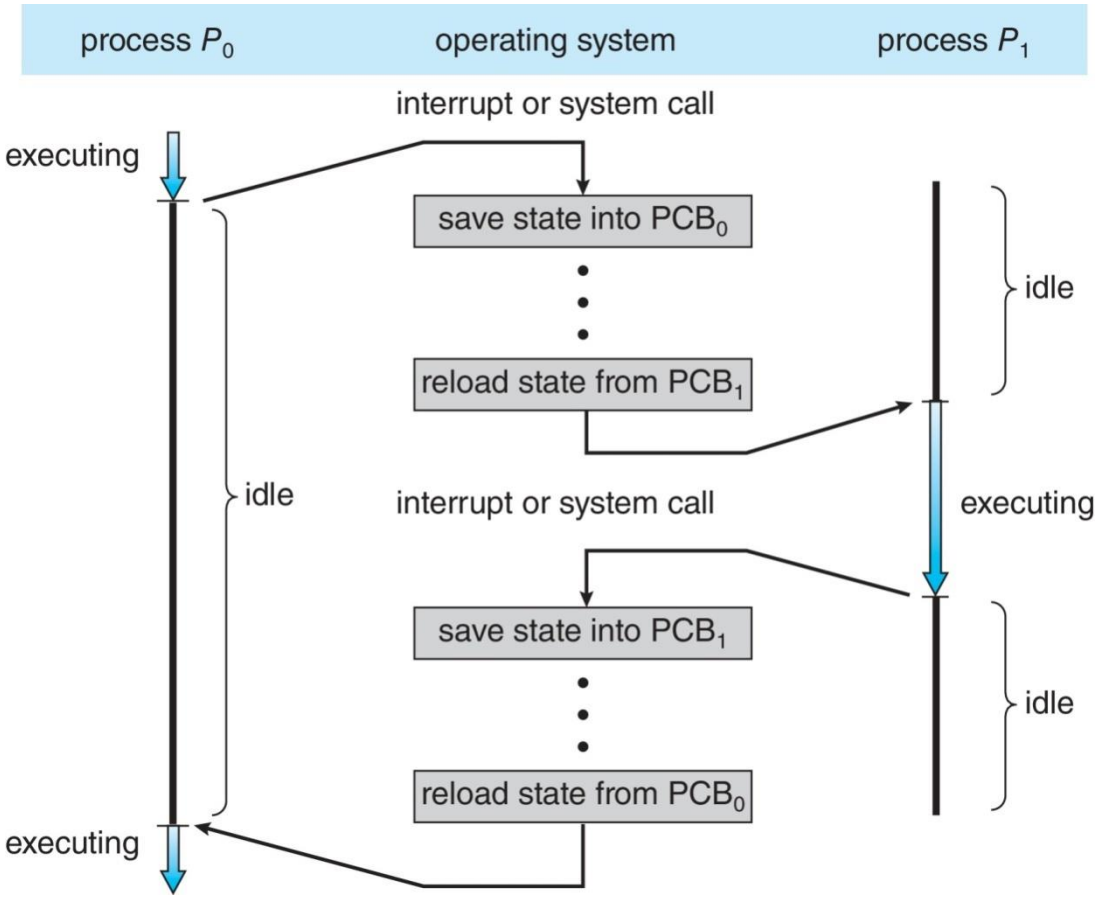
    In starting, all the processes get stored in the job queue. It is maintained in the secondary memory.
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device Queue(Wait queues)** – set of processes waiting for an event (i.e., I/O)
  - Processes migrate among the various queues

Ready and Wait Queues



CPU Switch From Process to Process

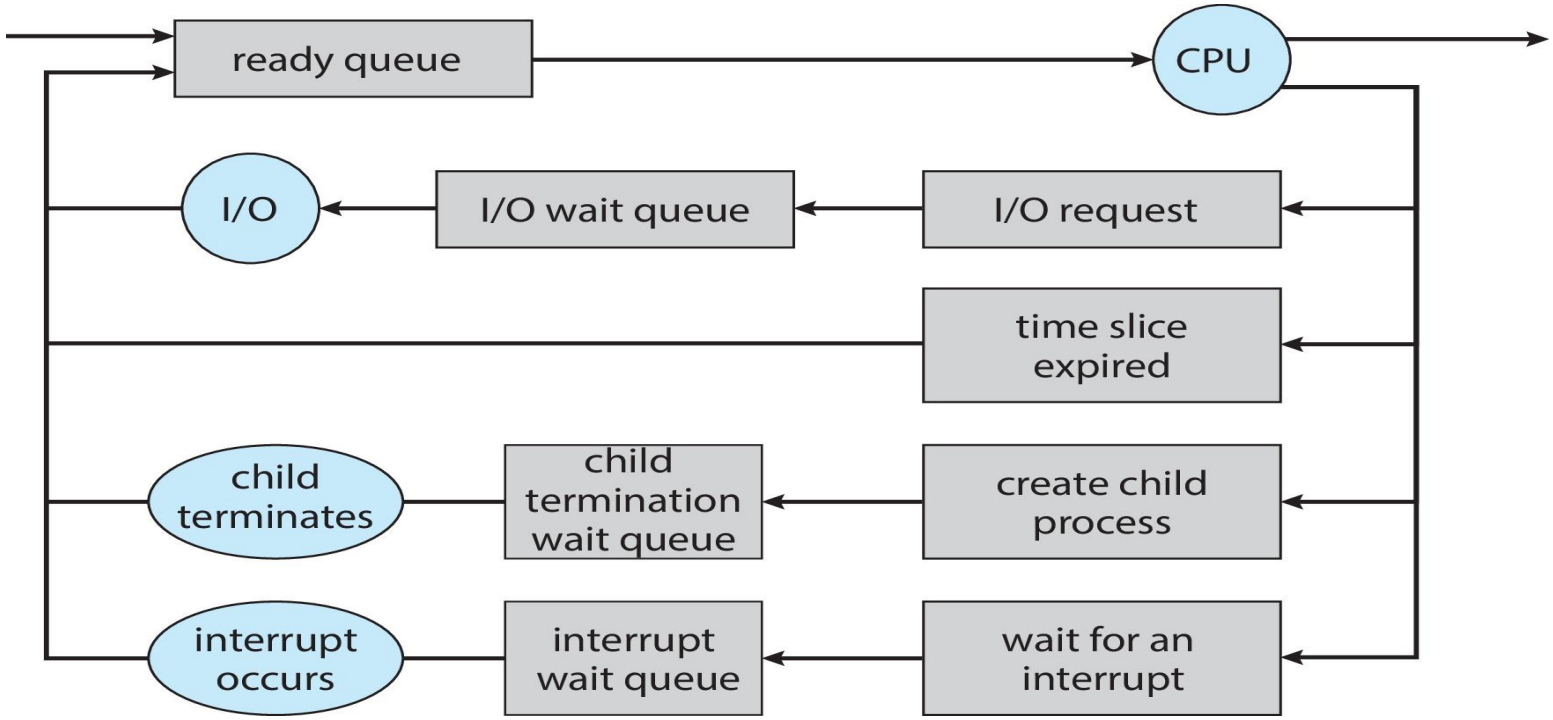
A **context switch** occurs when the CPU switches from one process to another.



Context Switch

- When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB
- Context-switch time is pure overhead; the system does no useful work while switching
  - The more complex the OS and the PCB ➔ the longer the context switch
- Time dependent on hardware support
  - Some hardware provides multiple sets of registers per CPU ➔ multiple contexts loaded at once

Representation of Process Scheduling

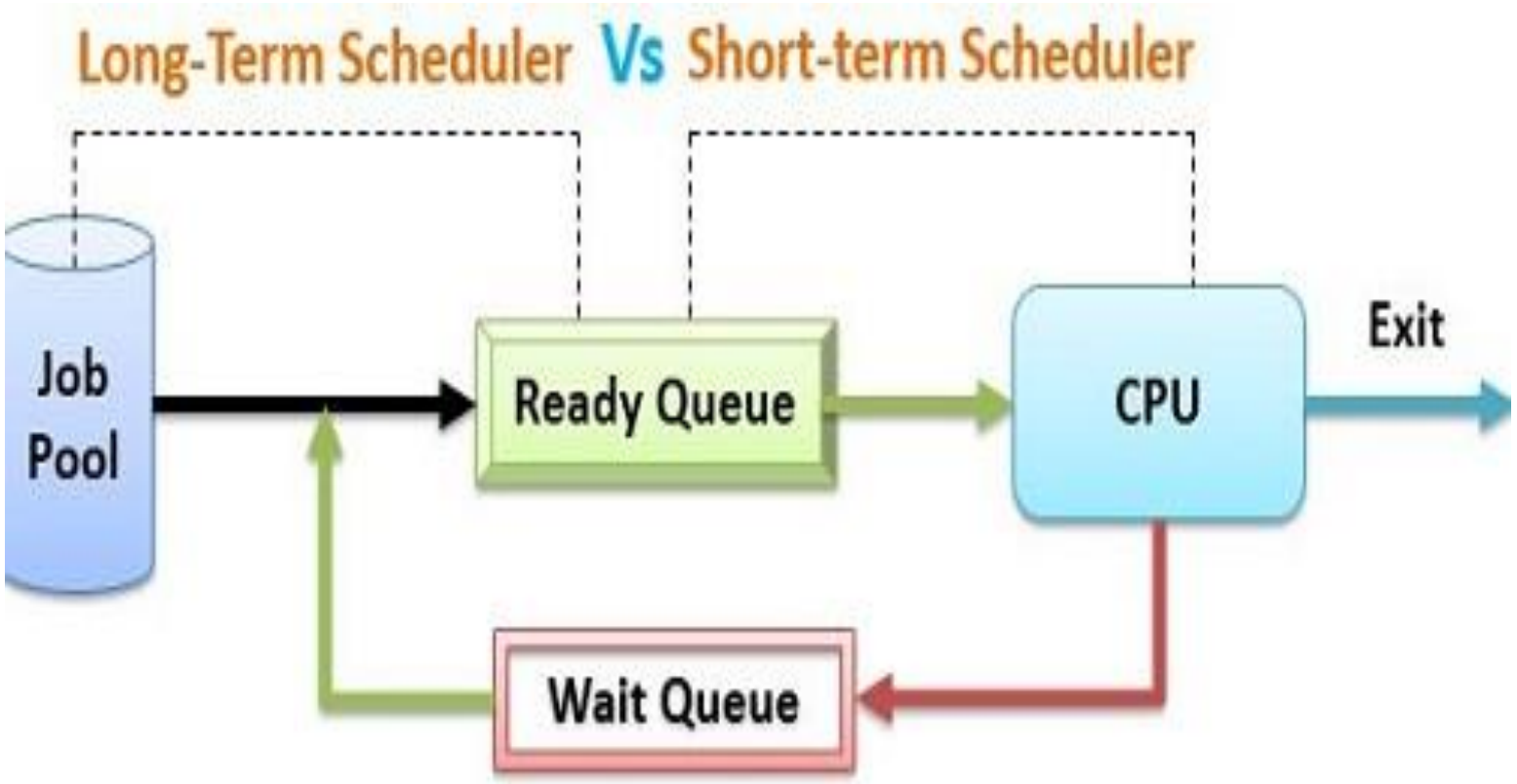


Schedulers

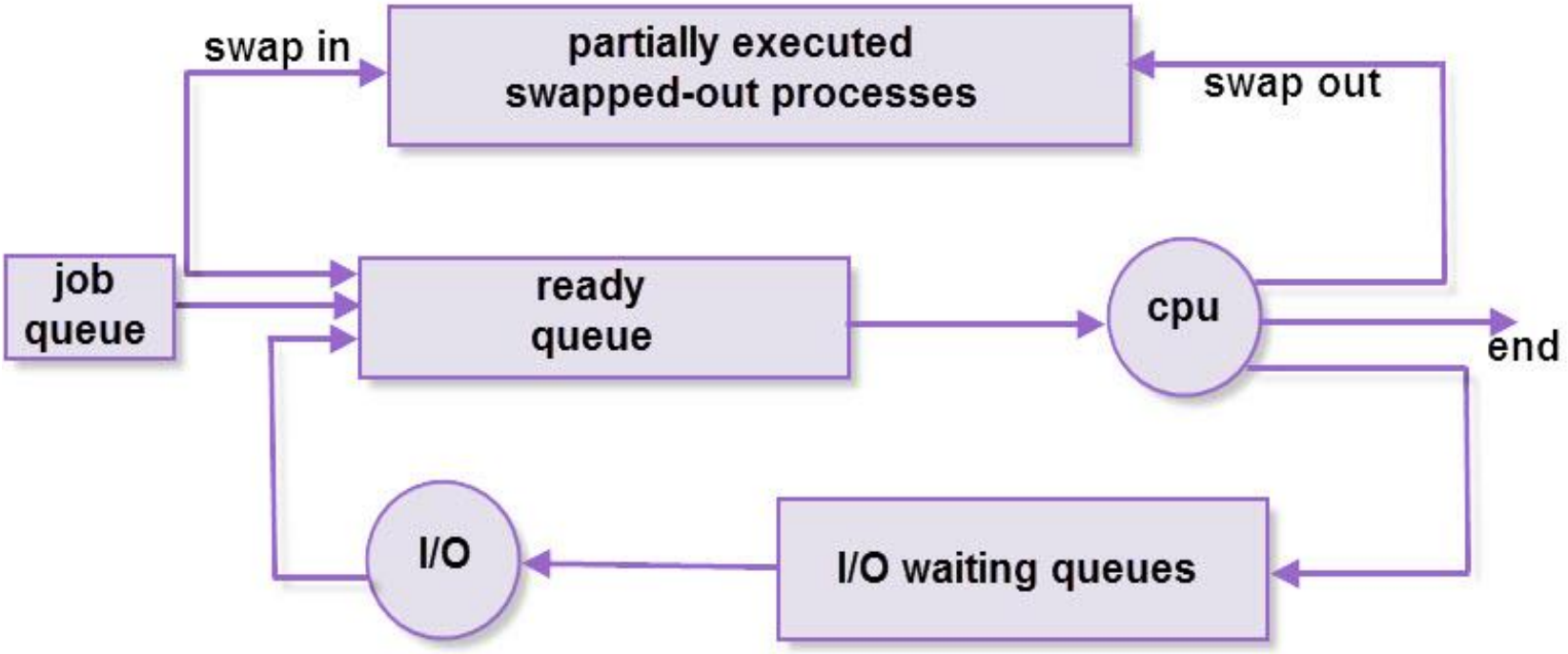
Schedulers are special system software which handles process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long-Term Vs Short Term Scheduler

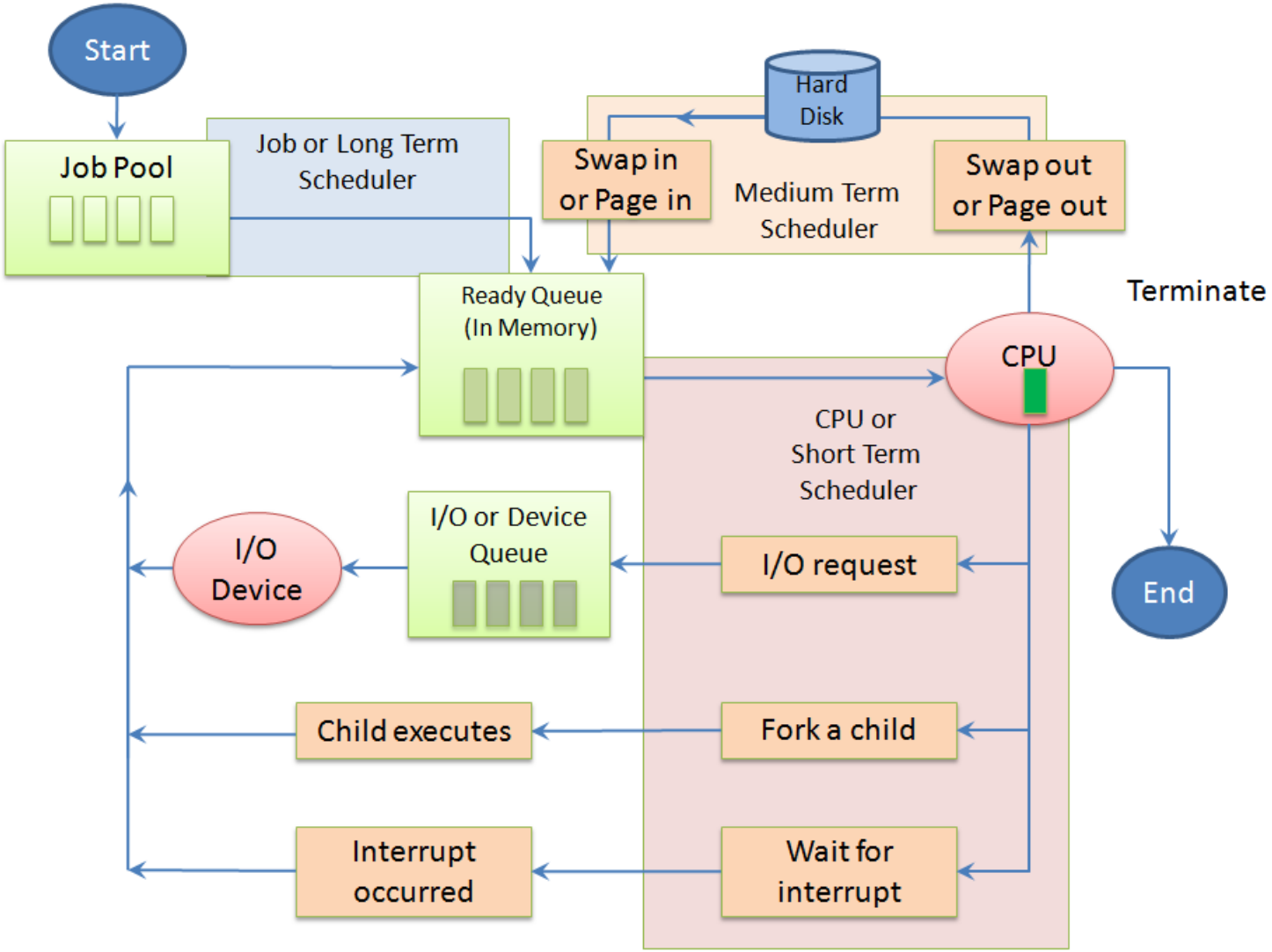


Medium Term Scheduler

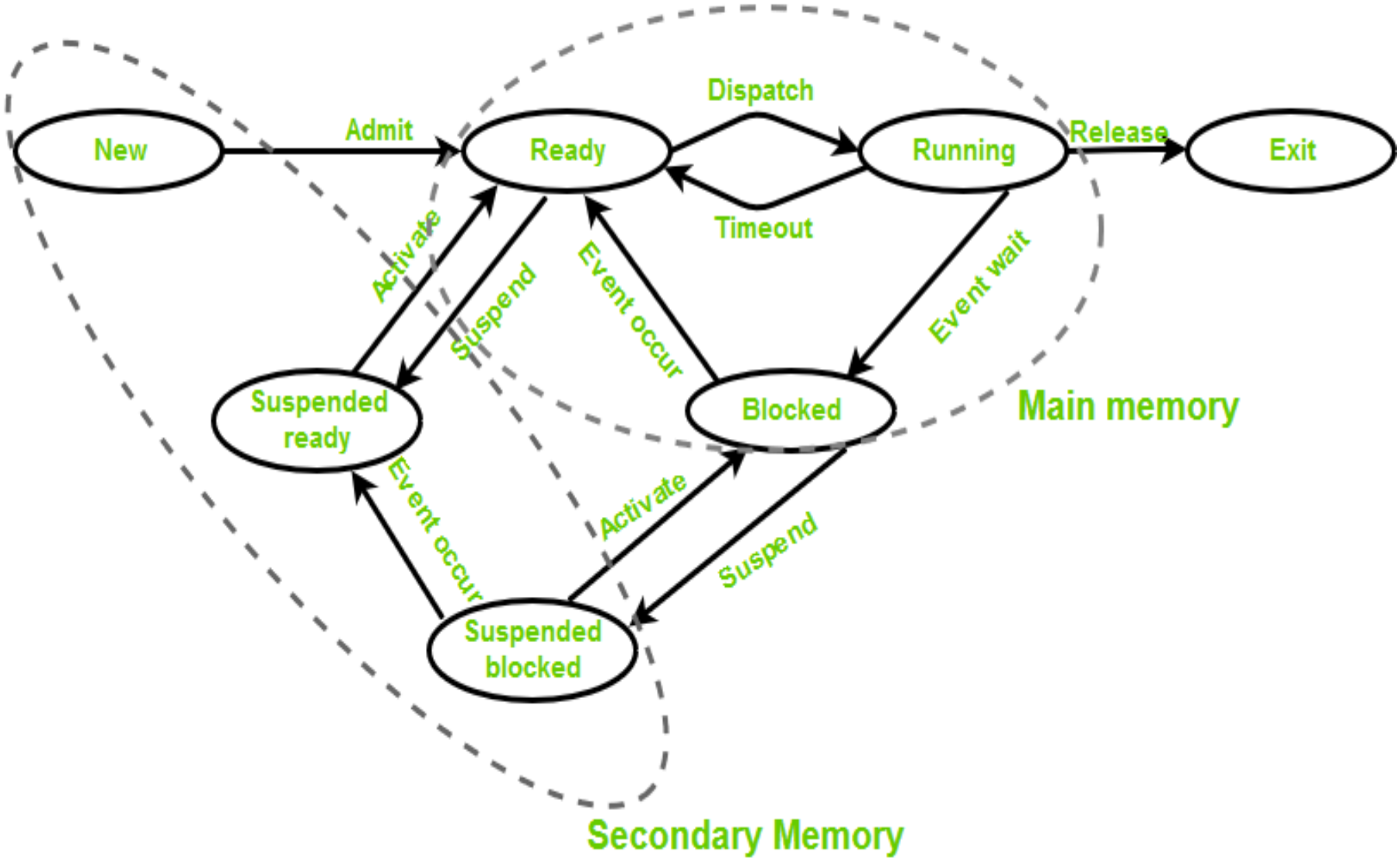


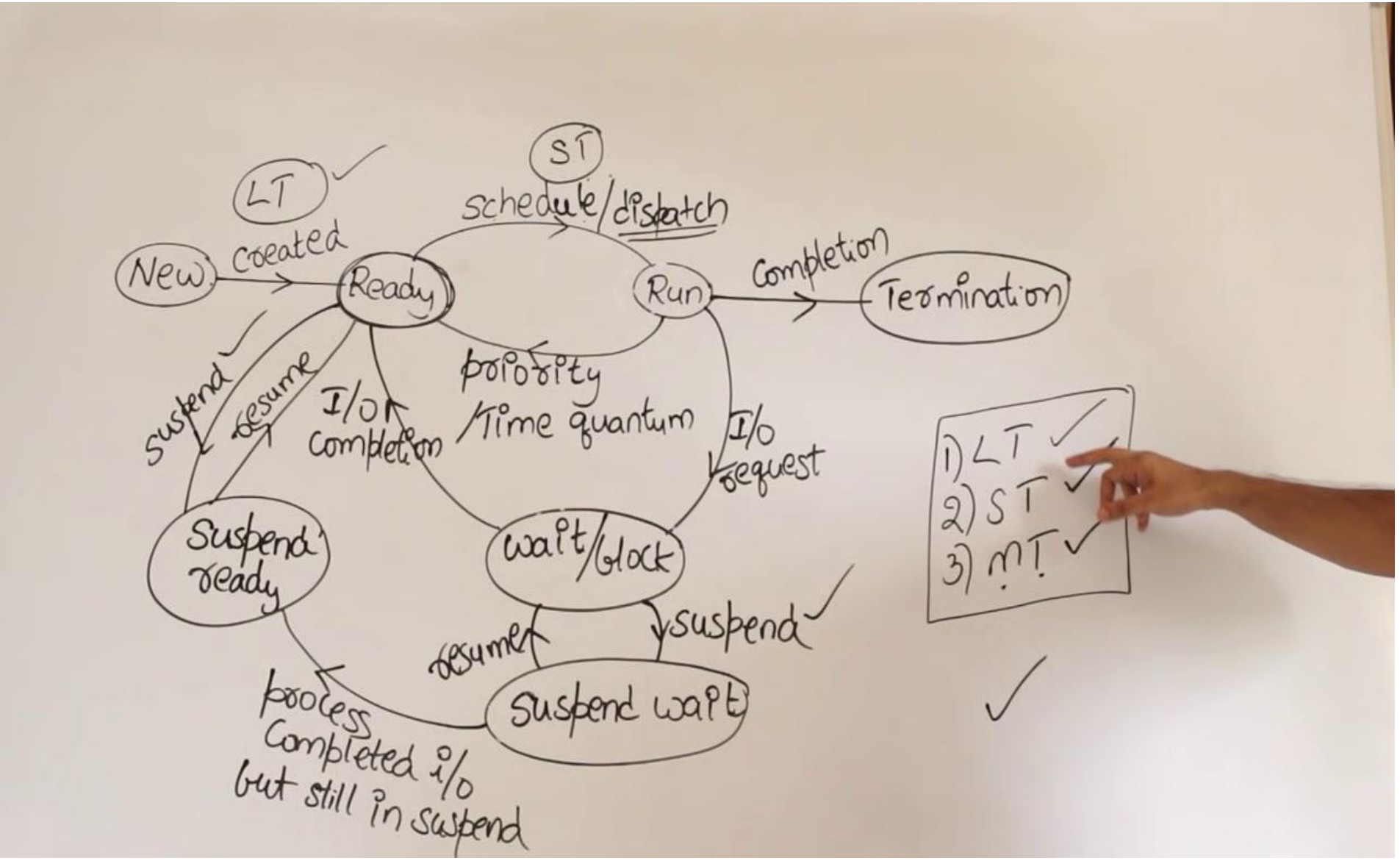
Addition of medium-term scheduling to the queuing diagram

Process Scheduling Queues



Process State Diagram (7 States)




LT
ST
schedule/dispatch
New
created
Ready
Run
Completion
Termination
suspend
resume
I/O completion
priority
Time quantum
I/O request
Suspend ready
wait/block
suspend
resume
process Completed i/o but still in suspend
Suspend wait
1) LT
2) ST
3) MT

Thank you

- https://www.javatpoint.com/os-process-states