PROCESSOR SCHEDULING PART I

2.1 Processes

- A **process** is an executing program, including the current values of the program counter, registers, and variables.

- The subtle difference between a process and a program is that the program is a group of instructions whereas the process is the activity.

2.1 Processes

- In multiprogramming systems, processes are performed in a pseudo parallelism as if each process has its own processor.
- In fact, there is only one processor but it switches back and forth from one process to another.

2.1 Processes

- Henceforth, by saying *execution* of a process, we mean the processor's operations on the process like changing its variables, etc.
- *I/O work* means the interaction of the process with the I/O operations like reading something or writing to somewhere.
- They may also be named as "**processor (CPU) burst**" and "**I/O burst**" respectively

2.1 Processes

- According to these definitions, we classify programs as
  - **Processor-bound program:** A program having long processor bursts (execution instants).
  - **I/O-bound program:** A program having short processor bursts.

2.1 Processes

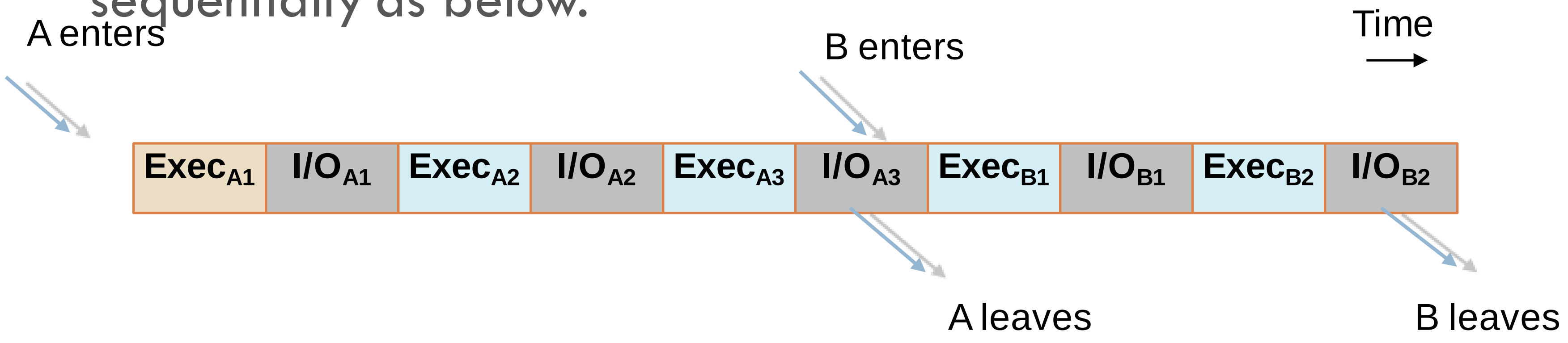
- Assume we have two processes A and B.
- Both execute for 1 second and do some I/O work for 1 second.
- This pattern is repeated 3 times for process A and 2 times for process B.

Exec$_{A1}$	I/O$_{A1}$	Exec$_{A2}$	I/O$_{A2}$	Exec$_{A3}$	I/O$_{A3}$

Exec$_{B1}$	I/O$_{B1}$	Exec$_{B2}$	I/O$_{B2}$

2.1 Processes

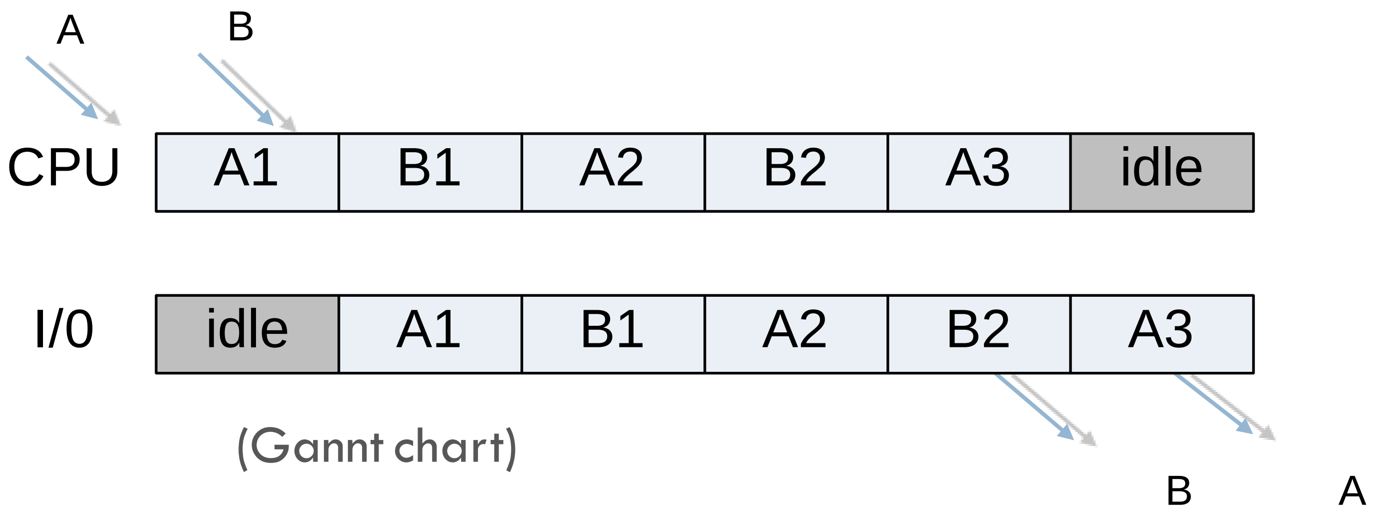
- If we have **no multiprogramming**, the processes are executed sequentially as below.



- So, the processor executes these two processes in a total time of 10 seconds. However, it is idle at I/O instants of processes. So, it is idle for 5 seconds and utilized for 5 seconds.
- Then the processor utilization is $\frac{5}{10} \times 100 = 50\%$

2.1 Processes

Now let's consider **multiprogramming** case:



(Gannt chart)

- In this case, when process A passes to some I/O work (i.e. does not use the processor), processor utilizes its time to execute process B instead of being idle.
- Here the processor utilization is $\frac{5}{6} \times 100 = 83\%$

2.2 Process States

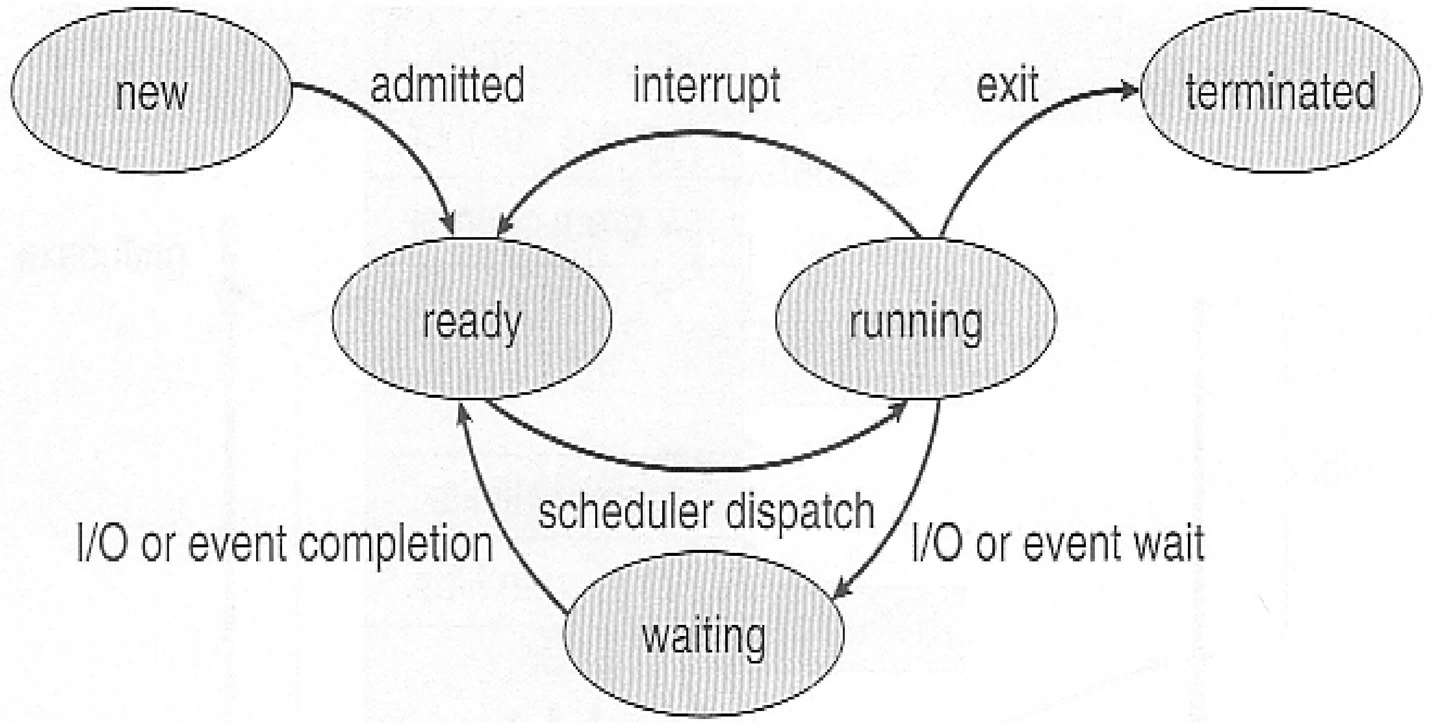


Figure 3.2 Diagram of process state.

2.2 Process States

- In the OS, each process is represented by its **PCB (Process Control Block)**. The PCB, generally contains the following information:

Process State
Process ID,
Program Counter (PC) value,
Register values
Memory Management Information (page tables, base/bound registers,…)
Processor Scheduling Information(priority, last processor burst time etc.)
I/O Status Info (outstanding I/O requests, I/O devices held, etc.)
List of Open Files
Accounting Information

2.2 Process States

- If we have a single processor in our system, there is only one running process at a time. Other ready processes wait for the processor.
- The system keeps these ready processes (their PCBs) on a list called **Ready Queue** which is generally implemented as a linked-list structure.
- When a process is allocated by the processor, it executes and after some time it either finishes or passes to waiting state (for I/O).
- The list of processes waiting for a particular I/O device is called a **Device Queue.**
- Each device has its own device queue.

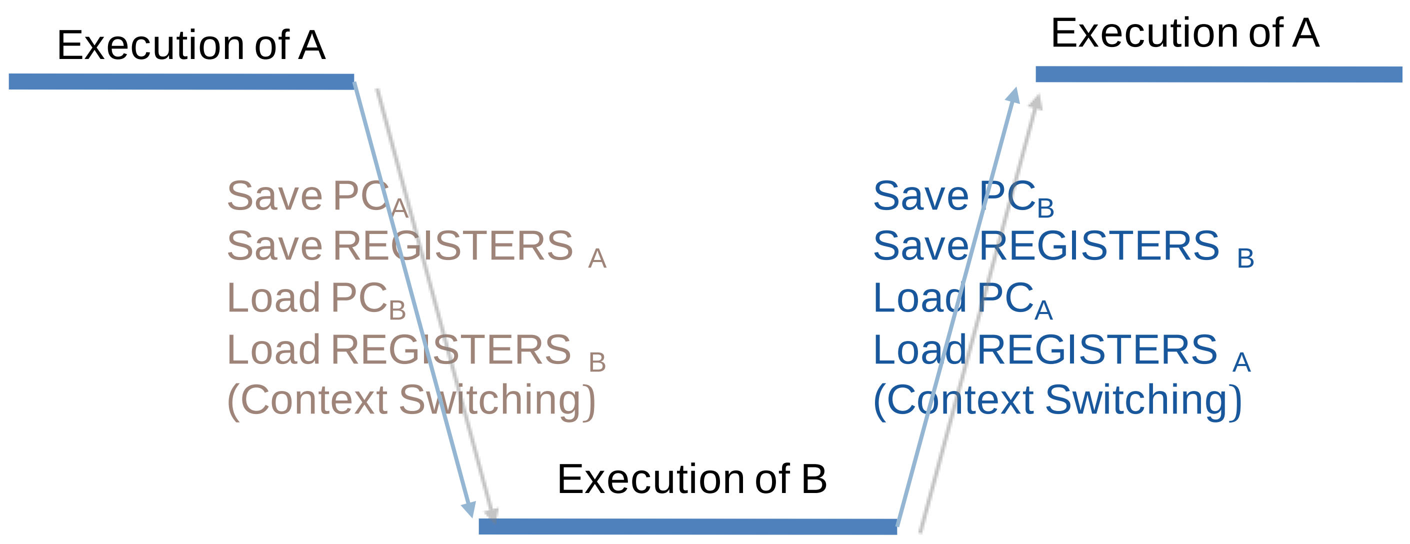
2.2 Process States

- In multiprogramming systems, the processor can be switched from one process to another.

- Note that when an interrupt occurs, PC and register contents for the running process (which is being interrupted) must be saved so that the process can be continued correctly afterwards.

2.2 Process States

- Switching between processes occurs as depicted below:



2.3 Scheduler

- If we consider batch systems, there will often be more processes submitted than the number of processes that can be executed immediately.

- So incoming processes are spooled (to a disk).

- **The long-term scheduler** selects processes from this process pool and loads selected processes into memory for execution.

2.3 Scheduler

- **The short-term scheduler** selects the process to get the processor from among the processes which are already in memory.
- The short-time scheduler will be executing frequently (mostly at least once every 10 milliseconds).
- So it has to be very fast in order to achieve a better processor utilization.
- The short-term scheduler, like all other OS programs, has to execute on the processor.
- If it takes 1 millisecond to choose a process that means (1 / (10 + 1)) = 9% of the processor time is being used for short term scheduling and only 91% may be used by processes for execution.

2.3 Scheduler

- The long-term scheduler on the other hand executes much less frequently.
- It controls the degree of multiprogramming (no. of processes in memory at a time).
- If the degree of multiprogramming is to be kept stable (say 10 processes at a time), then the long-term scheduler may only need to be invoked when a process finishes execution.

2.4. Performance Criteria

- In order to achieve an efficient processor management, OS tries to select the most appropriate process from the ready queue.
- For selection, the relative importance of the followings may be considered as performance criteria.

2.4. Performance Criteria

- **Processor Utilization:**

  The ratio of busy time of the processor to the total time passes for processes to finish.

$$\text{Processor Utilization} = \frac{\text{Processor busy time}}{\text{Processor busy time} + \text{Processor idle time}}$$

  - We would like to keep the processor as busy as possible.

2.4. Performance Criteria

- **Throughput:**

  The measure of work done in a unit time interval.

$$Throughput = \frac{Number\ of\ \ processes\ completed}{Unit\ time}$$

2.4. Performance Criteria

- **Turnaround Time (tat):**

  The sum of time spent in the ready queue, execution time and I/O time.

  tat = t(process completed) – t(process submitted)

2.4. Performance Criteria

- **Waiting Time (wt):**

  Time spent only in ready queue.

  Processor scheduling algorithms only affect the time spent waiting in the ready queue. So, considering only waiting time instead of turnaround time is generally sufficient.

2.4. Performance Criteria

- **Response Time (rt):**

  The amount of time it takes to start responding to a request. This criterion is important for interactive systems.

$$rt = t(first\ response) - t(submission\ of\ request)$$

the time that the process starts execution

2.4. Performance Criteria

- We, normally, want to maximize the processor utilization and throughput, and minimize tat, wt, and rt.
- However, sometimes other combinations may be considered depending on the system requirements.
- For example, for the interactive time sharing systems, response time is quite important.