الملزمة الثانية

Process Management

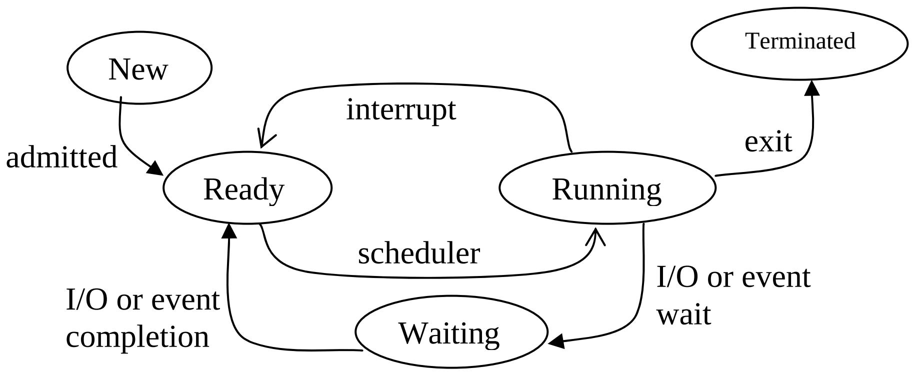
- a *process* is a program in execution, which needs *resources* to accomplish its task.
- Process types: *1. O.S processes 2. user processes.*
- Ex: word program, compiler, printer, web browser,…

Process States

As process executes, it changes its state. Each process may be in one of the following states:

1. ***new***: the process is been created.
2. ***Running***: instructions are being executed.
3. ***Waiting***: waiting for some event to occur (such as I/O completion)
4. ***Ready***: the process is waiting to be assigned to a processor.
5. ***Terminate***: the process has finished its execution.

Note: only one process can be running at any instance. Many processes may be ready or waiting.

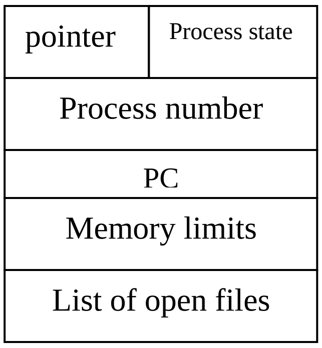


[Diagram of process states]

Process Control Block (PCB)

- Each process is represented in the O.S by PCB. It consist the following information:

1. *program Counter*: to indicate the address of the next instruction to be executed for this process.
2. *CPU registers*: accumulator, index register, stack pointer, and general purpose registers, these registers must be saved when an interrupt occur, to allow the process to be continued correctly afterward.
3. *CPU-scheduling information*: such as a process priority, pointers to scheduling queues.
4. *Process State*: the state may be new, ready, running, waiting, and halting.
5. *Memory Management information*: such as page table or segment table.
6. *I/O information*: include list of I/O devices associated to this process and list of open files.



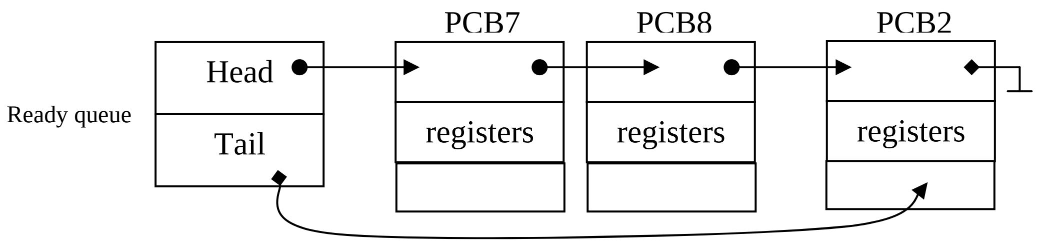
[PCB]

Process Scheduling

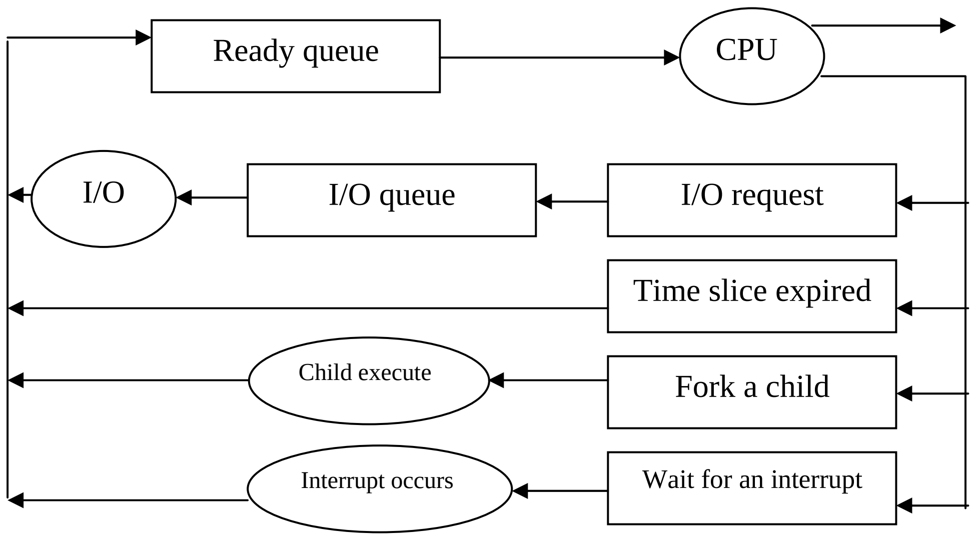
- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
- The objective of time sharing is to switch the CPU among processes so frequently.
- For a uniprocessor system, there will never be more than one running process. If there are more processes, the rest will have to wait until the CPU is free.

Scheduling Queues

- As processes enter the system, they are put into a ***job queue***. This queue consist of all processes in the system.
- The processes that are residing in main memory and are ready and waiting to execute are kept on list called ***ready queue***.



- The list of processes waiting for a particular I/O device is called ***device queue***. Each device has its own device Queue.
- A new process is initially put in the ready queue. It waits in the ready queue until it is selected for execution. <u>*Once the process is allocated the CPU and is executing, one of several evens could occurs*</u>:
  1. the process could issue an I/O request, and then be placed in an I/O queue.
  2. the process could create a new sub process and wait for its termination.
  3. the process could be removed from the CPU, as a result of an interrupt, and be put back in the ready queue.
     - A process continue this cycle until it terminates. Its then removed from all queues and its PCB and resources deallocation.



Queue diagram of process scheduling

Schedulers

- O.S select processes from queues in some fashion by the scheduler.
- Long- term scheduler [***Job scheduler***] select processes from disk (pool) and loads them into memory for execution.
- Short- term scheduler [***CPU scheduler***] selects from the process that are ready to execute, and allocate the CPU to one of them.
  - the short-term scheduler must select a new process from the CPU frequently.
  - The long-term scheduler executes less frequency.
  - long-term scheduler controls the [***degree of multiprogramming***]: number of processes in memory.
- There is two types of processes:
  1. *I/O bound process*: spend more of its time doing I/O.
  2. *CPU bound process*: spend more of its time doing computation.
- Its important that the long-term scheduler select a good (process mix) of I/O bound and CPU- bound process.
- Some O.S uses *medium-term scheduler,* to remove processes from memory, and thus to reduce 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***].

Cooperating processes

- A process is *independent* if cannot affect or be affected by the other processes in the system.
- A process is *cooperating* if it can affect or be affected by the other processes in the system.
- Many reasons for process cooperating:
  1. *information sharing.*
  2. *computational speedup.*
  3. *modularity*