Process Management

Process

A process or task in an instance of a program in execution. It is the smallest unit of work individually schedulable by an OS.

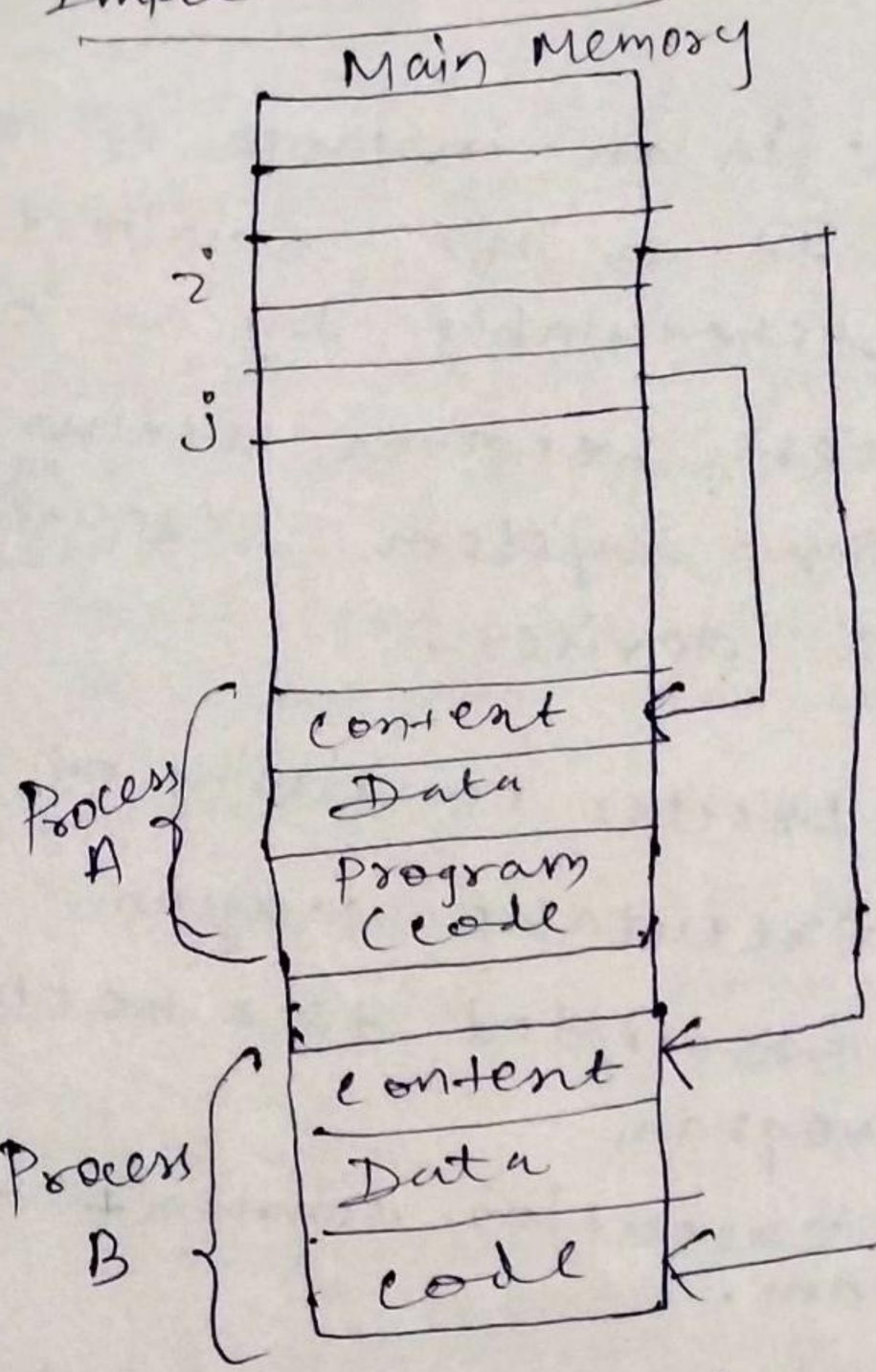
→ Once created a process becomes active & eligible to compete for system resources such as processor & I/O devices.

→ We can think of a process consisting of 3 components. 1) The executable program
2) The associated data needed by the program
3) The execution content of the program.

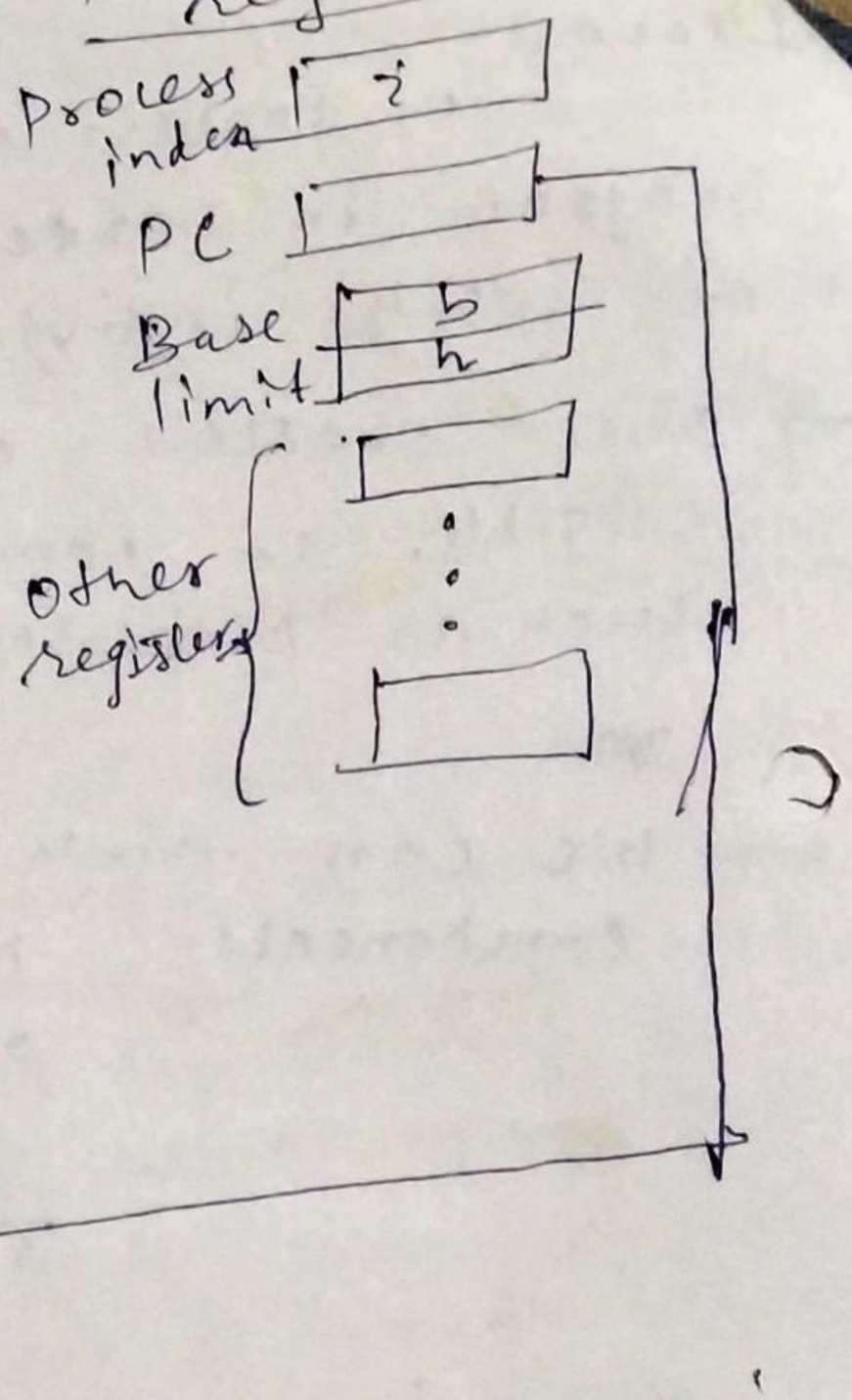
→ Execution content:- It includes all the infor^n that the OS needs to manage the process. It includes the contents of various processor registers such as Program counter holding the address of next instr^n, data registers etc, process stack containing temporay data, such as subroutine parameters, return addresses etc, other infor^n like process state, scheduling priority etc.

→ When the user initiates to execute a program, the OS responds to it by creating a process After the completion of execution, the OS deletes that process. The deletion of process doesn't affect original program or result. If another user invokes the same program, the OS responds to it by creating another process.

Process Implementation



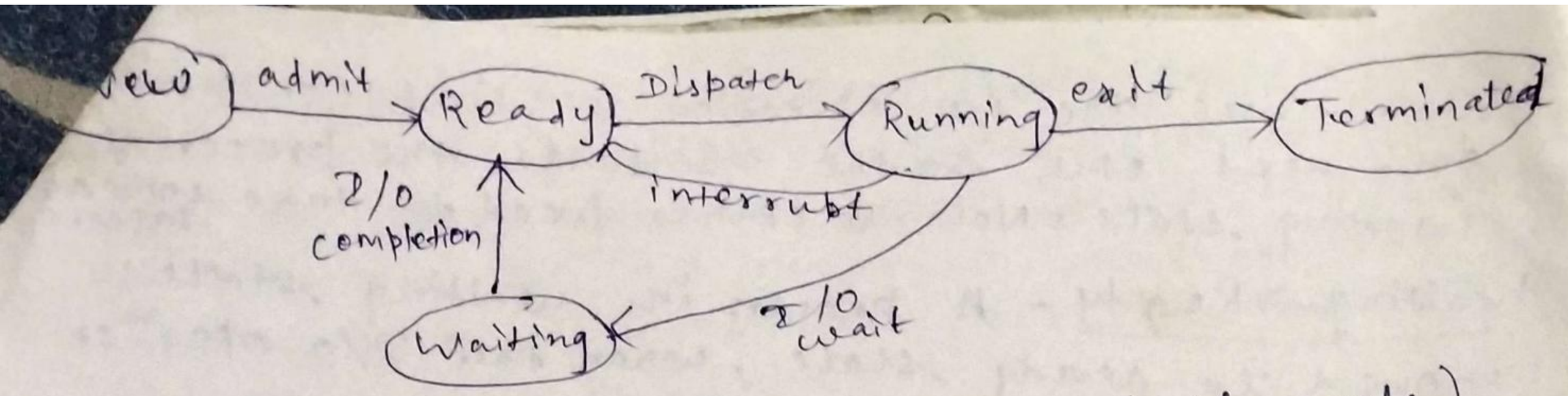
Process registers



Process states & life cycle

Generally the state of the process is determined by the current activity of the process. During execution the process changes its states. At any time, a process remains in one of the states mentioned below.

1. **New** – The process is being created.
2. **Running** – The " " " executed.
3. **Waiting** – " " " waiting for some event to occur.
4. **Ready** – The process is waiting to be assigned to a processor i.e it is ready for execution & all the resources except the processor has been assigned to it.
5. **Terminated** – The process has finished execution



(Process state diagram/ life cycle)

New → Ready — The OS creates a process & prepares the process to get executed. when the process gets ready & acquires various resources, the OS moves the process into Ready queue (a queue where all ready process remains)

Ready → Running — The OS selects one of the process from the ready queue, assigns processor to it & moves the process from ready state to running state to do execution.

Running → Terminated — When the execution of a process has been completed, the OS terminates the process & moves it from running to terminated state. Sometime processes are terminated due to time limit exceeded, memory unavailable, protection error or due to some other erroneous condⁿ.

Running → ready — The running process is preempted. (CPU is forcefully taken away from it) if some higher priority process arrives or the time slot expired or the processer receives any interrupt signal; then the OS shifts running process to ready state.

Running → Waiting → A process is put into the waiting state, if the process needs an I/O

oper^n. Since the I/O operation is time consuming & doesn't need CPU, so the OS sends the process to waiting state. Now the CPU is freed to take some other process.

Waiting → Ready - A process in waiting state moved to ready state, when the I/O oper^n or event has been completed.

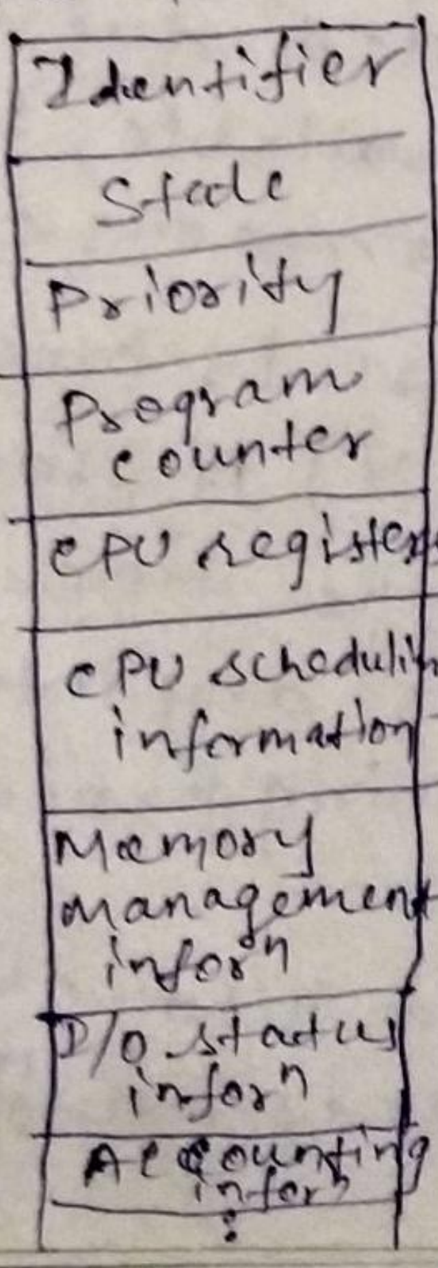
Process control Block (PCB)

⇒ The OS groups all info^n that it needs about a particular process (i.e execution context) into a specific data structure called PCB.

→ Whenever a process is created, the OS creates its corresponding PCB. When the process terminates its corresponding PCB is also released.

→ A process becomes known to the OS & becomes eligible to compete for system resources only when it has an active PCB associated with it.

→ The PCB consists of the following information



1) Process Id - An unique identifier is associated with an process to distinguish it from all other processes.

2) State - It tells that the process currently lies in which state.

3) Priority - It indicates the priority level of process w.r.t other

4) PC - It indicates the address of the next instr^n to be executed for the specific process

5) CPU registers - These registers vary in no. & type depending upon computer architecture. They include

cumulators, index registers, stack pointers & other general purpose registers. These are used to store various ~~and~~ state infor^n, during the occurance of an interrupt, so that the process will be continued correctly after its return.

CPU scheduling infor^n :— This infor^n includes the process priority infor^n, pointers to the scheduling queues & other scheduling parameters

Memory Management infor^n :— This infor^n includes the values of base & limit registers, page tables, address of program code & data associated with the process, memory blocks shared with other processes etc.

I/O status infor^n :— It includes the list of I/O devices & files allocated to the process

Accounting infor^n :— It includes the amount of processor time & real time (clock) used by the process, time limits, account nos, process nos etc.

Scheduling

Scheduling refers to a set of policies & mechanisms of the OS, that governs the order in which the work to be done by a computer system is completed. In simple terms it is the process of determining which process will actually run, when there are multiple processes to run.

Types of Scheduling

There are 2 types of scheduling
1) Non-Preemptive 2) Preemptive

(1) Non-preemptive - Once the CPU is assigned to a process, the CPU will not release it until its completion.

(2) Preemptive - Here CPU can release the process even in the middle of execution. For eg:- if a process with higher priority becomes ready for its execution, the process which is currently using the CPU, will be forced to give up the CPU, so that the higher priority job can execute first.

Non-Preemptive	Preemptive
(1) Once the process is allocated to CPU, CPU can't be taken away from it before its completion	(1) CPU can be taken away before the completion of a process
(2) No preference to a higher priority job	(2) Preference to a higher priority job.
(3) All the processes are treated equally.	(3) Not fair to all the processes.
(4) Cheaper Eg - FCFS scheduling algh	(4) costlier. Eg - Round Robin scheduling algh.

Problem in preemptive scheduling

When two processes share some data & one is updating the data & in the mean time, the 2nd process preempts it, then the 2nd

process might access in consistent data left by the 1st process.

Scheduling Queues

Job Queue

As processes enters the system, they are put into a job queue. This queue consists of all the processes in the system.

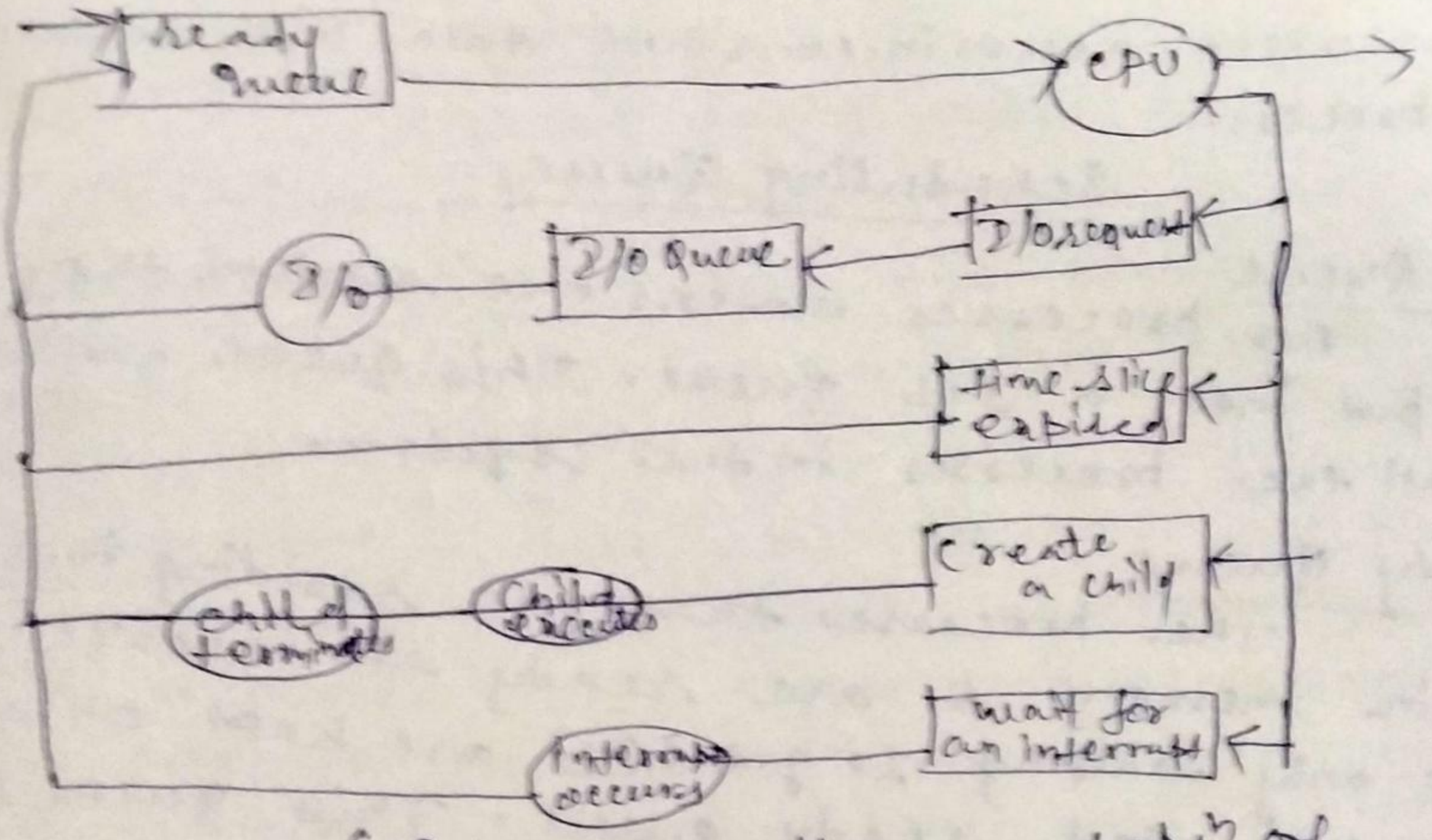
Ready Queue

The processes that are residing in the main memory & are ready to execute (i.e only waiting to get CPU) are kept on a queue called ready queue. This queue is implemented as linked list.

Device queue :-

Suppose a process is allocated to CPU & it is executing. In the meantime some interrupt occurs & the process has to perform some I/O related task, then such a process must be dedicated to the corresponding I/O device. But the I/O device may be busy in handling some other process's I/O operation. Therefore the process must have to wait for that particular I/O device. Such processes are kept in the device queue. Each I/O device has its own device queue.

In the following figure, the circles represent the resources that provides services to the queue, the arrow indicates the flow of processes & the rectangular box represents the queue (ready queue & set of device queues)



(Queueing diagram repsⁿ of process scheduling)

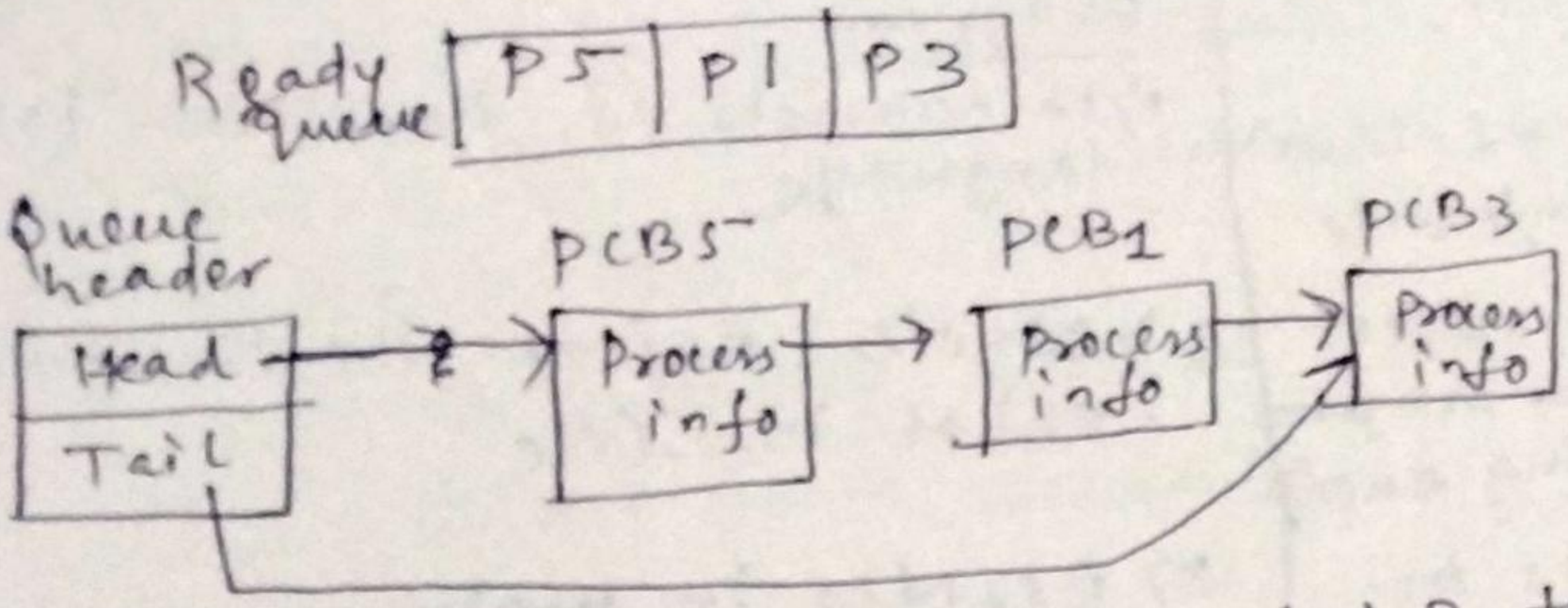
→ During execution, several events could occur. a) If the process issues an I/O request, then it can be placed in an I/O queue. After completion, it can again placed in ready queue.

b) If the time slice of a process expires, then such process can be placed back in ready queue.

c) The process may create a child process or sub process & wait for its termination. In this case also the process can move from waiting to ready state & put back to ready queue.

d) the process could be forcibly removed from the CPU due to some interrupt & after its completion, it may put back to ready queue.

Implementation of Ready Queue in linked list



→ Queue header points to the 1st & tail points to the last.

→ Each PCB has a pointer field that points to the next process in the ready queue.

Differentiate both process & program

Process	Program
1) It consists of instrn exec'n in m/c code	1) It consists of instrn in prog. language
2) Dynamic object i.e it is the program loaded during exec'n	2) Static object i.e it only exist in file
3) It resides in the ~~sec. storage devices~~ main memory	3) Resides in ~~main memory~~ secondary storage devices
4) ~~Passive entity on disk~~	~~5) Active entity i.e acti.~~
4) Active entity i.e actively executing	4) Passive entity on disk
5) Time span is limited	5) Time span is unlimited.
6) It contains more than the program code including execution content.	6) It is only a part of process