Process Management

Process is a program in execution

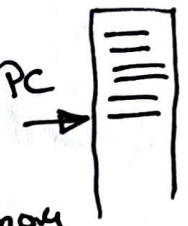
Composed of:

text section: program code

activity: get ID.

Program Counter

Registers.



heap (dynamic memory
new malloc).

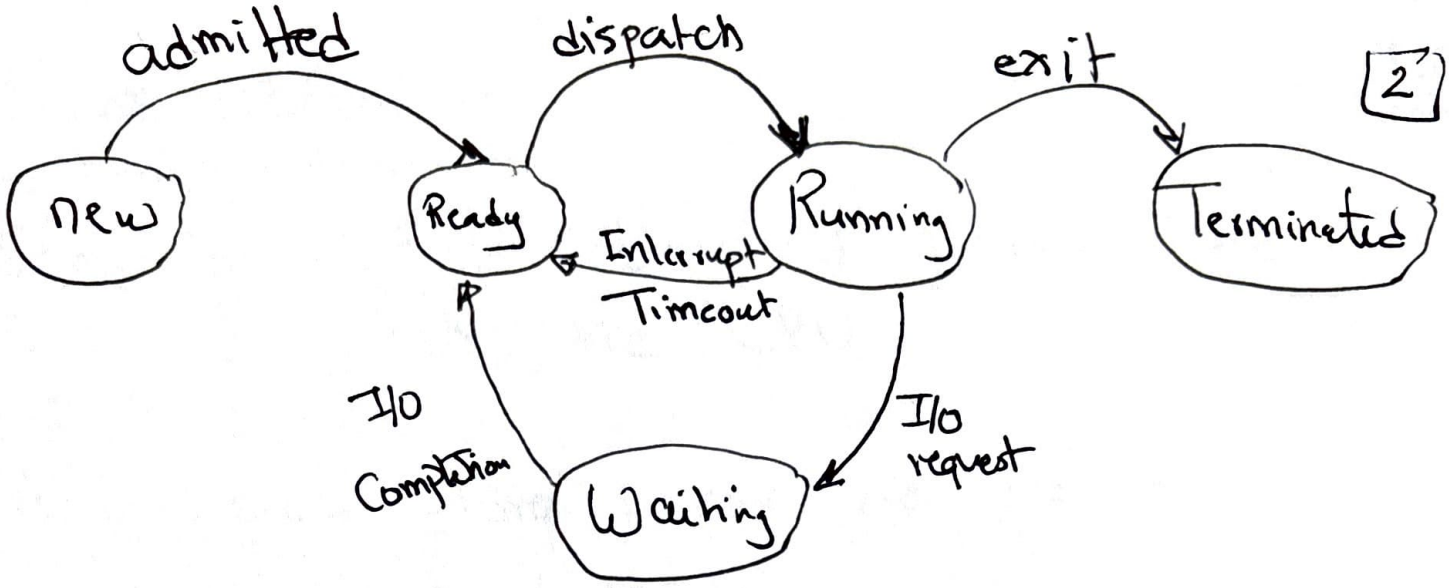
Stack (temp. data).

global variables (data section).

Process State Diagram

As a process executes, it changes its state.

1. New: being created.
2. Running: executing instructions on the CPU.
3. Waiting / Blocked: waiting for an event to happen (I/O).
4. Ready: waiting for the CPU.
5. Terminated: has finished.



Process ControlBlock (Task ControlBlock).

Each process is represented in the OS by a (PCB)

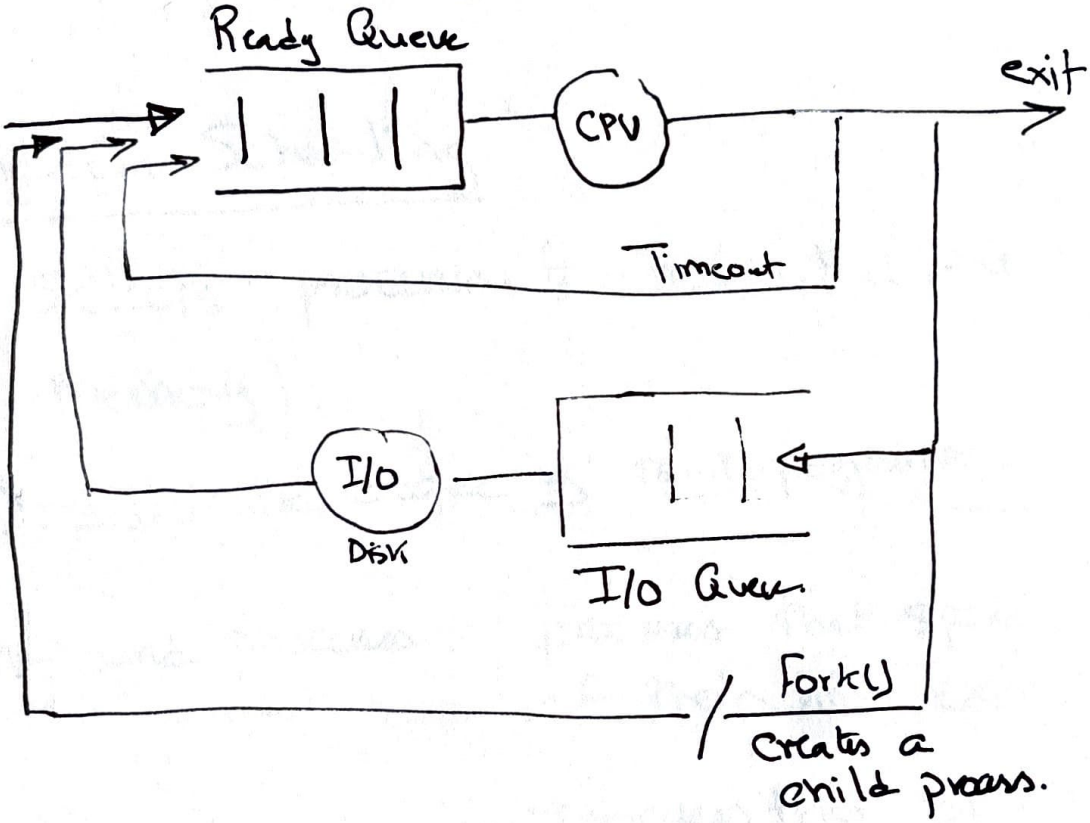
PCB ≡ data structure.

- ID
- State
- PC + registers (Stored / loaded when interrupt happen and to resume).
- Accounting info (User , path , time used on the CPU, ...)
- Memory pointers
- Priority for scheduling
- Threads.
- I/O status information (list of open files/devices)

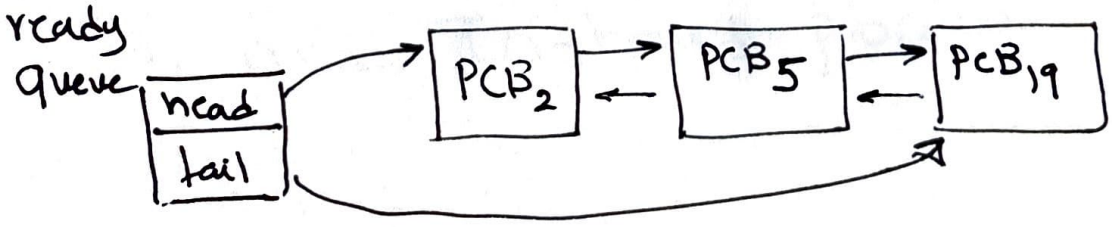
Process Scheduling

Scheduler : select the next process to the use the CPU.

Ready Queue: stores process that are in the Ready state.



Qeues are generally stored as Linked lists.



A process moves between different queue during its life time.

Short-term Scheduling

Select one of the ready processes to run on the CPU.
(executed more frequently)

Long-term Scheduling

Selects processes to admit (put in memory).

Decides the degree of multiprogramming

CPU-bound processes: processes that spend most of their time executing

I/O-bound processes: processes that spend most of their time issuing I/O.

It's good to have the right mix of CPU and IO-bound processes.

Medium-term Scheduler

Removes process from memory to disk (swapping).

Context Switching:

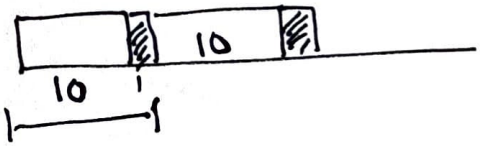
Switching the CPU to another process.
(involves state save and state restore).

Pure Overhead

It takes 1 msec to run the scheduler to pick a process to run for 10 msec.

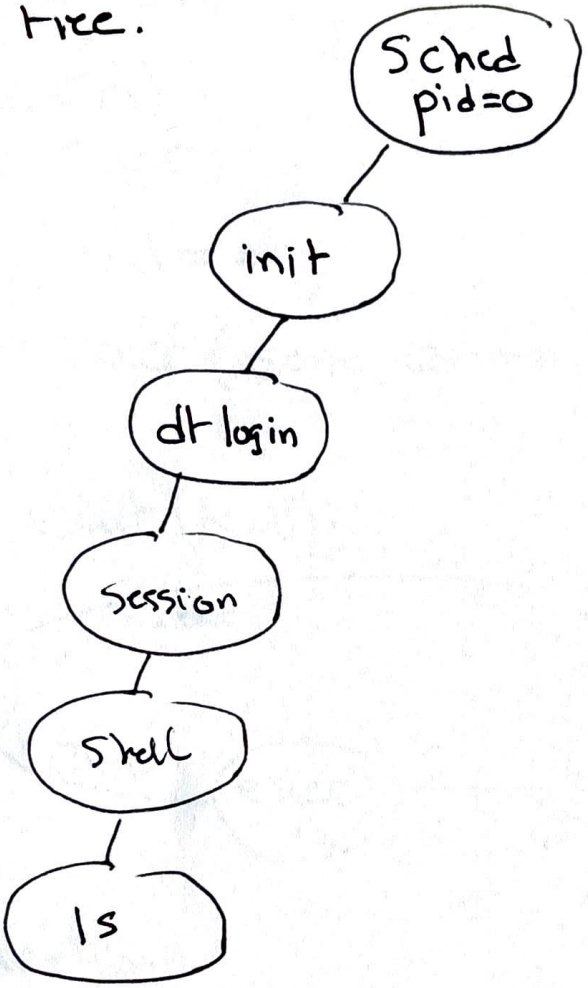
What's the overhead?

$$\frac{1}{10} = 9\%.$$



Process Creation

1. OS creates first process.
   - Forms a tree.



A process can create another process

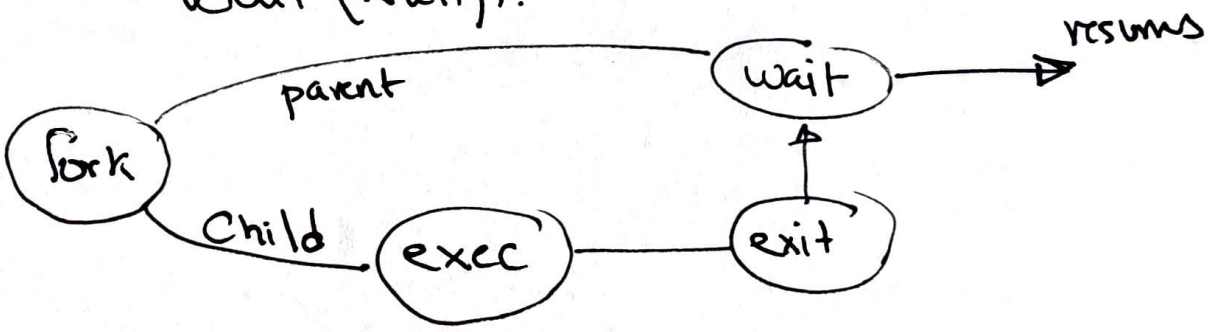
parent → child

When a child process is created

1. Parent can continue running
2. " waits for the child to finish

A new child process is created with fork

```
pid = fork();
If (pid < 0)
        error;
elseif ( pid == 0)
        exec( some command)
else
        wait (Null);.
```



Process Termination

- exit();
- Parent terminating
- errors.