Introduction To Process

* Program → Compiled Code Ready to Execute
* Process → Program under Execution.
* Current Status of Process is indicated by PC (Program Counter) and CPU registers.

→ Process Structure :-

.cpp (file) → Compiler (given to) → Compile (it) → Executable (make it) → Program.

*) Process Memory : It is divided into 4 Sections.

i) Stack ii) Heap
iii) data iv) text.

Architecture → "of Process in Memory" __/__/____

Max

Stack	→ It is used to Store local Variables, fun^n Parameters, fun^n Values, return value & address etc ^...n
↓ (overflow) / (out of memory) ↑	* This Space grows when Stack is full (downward) & vice versa.
Heap	→ "allocating memory without free" → used for dynamic memory allocation
data	→ Store global & Static Variables
text	→ Comprises Compiled Program Code. (load from disk). goal is to "Execute this Code".

Attributes of Process :-

a) Features that allows identifying a Process uniquely.

b) Process table :

↳ All Process are being tracked by OS using table like data Structure Called "PCB".

↳ each entry in that table is : PCB.

↓

"Stores info /attributes of a Process".

*) TikTok → Program → O.S → Convert to Process.

*) Why Program → To help users to Communicate with O.S & PC.

4Q) How O.S Creates Process :.

Ans) By Converting Program into Process via 5 Steps;

i) Load the Program and Static data to memory.

(Initializing data).

ii) Allocate Runtime Stack.

(Part of memory used for local variable, return value ....)

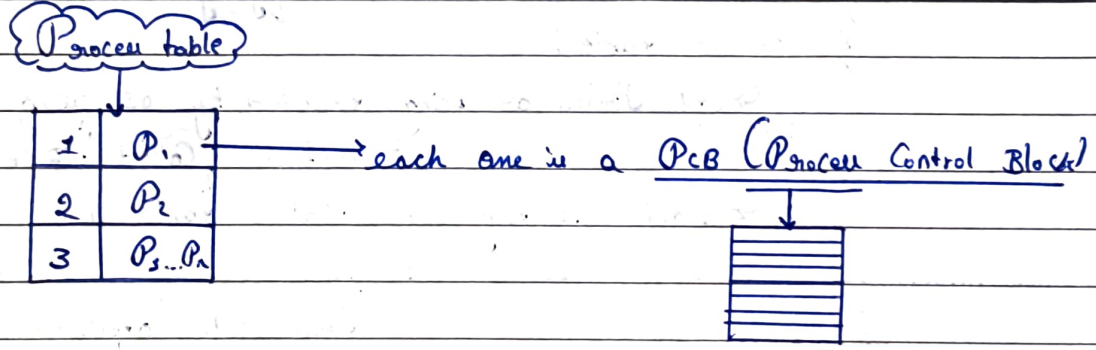
__/__/____

iii) Heap memory allocation : (Allocating Runtime Stack).
↳ Part of memory used for dynamic Allocation

iv) To tasks :
(Unix) → file, ole and error handlers are allotted.

v) O.S <handoffs Control to main(). (O.S know only main)
(returns 0 -> O.S <believe program Executed Successfully).

- **Stack overload :-** (overflow)
  Avoid : Stack unwinding should be done. Setting a limit for our recursion fun^n. (Base Case set)
- **Out of Memory :**
  Avoid : deallocating all the unnecessary memory alloted.



- **Registers in CPU :-**
  * It is a data Structure
  * When a Process is running and its time Slice Expires, the Current value of Process Specific Register would be Stored in PCB and would be Snuffed out.
  * When the Process is Scheduled to be run, the register Value is read from PCB and written to CPU register.
    ↳ "This is the main purpose of registers in PCB".

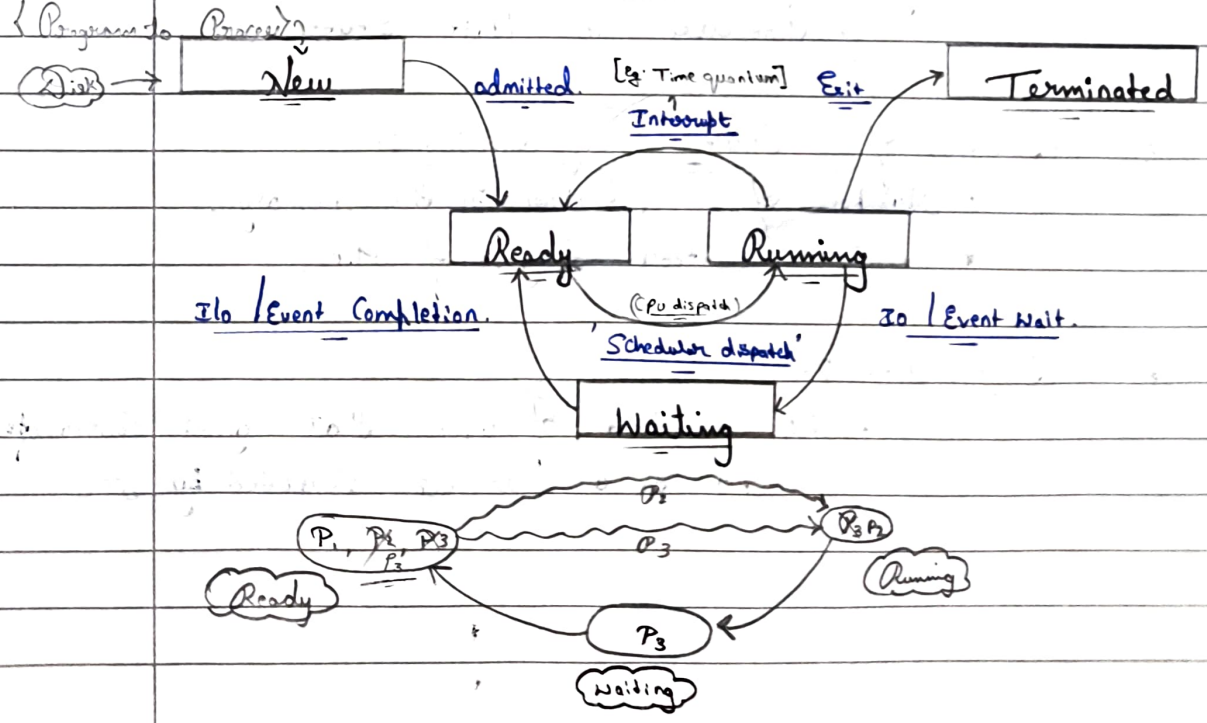
__/__/____

Process States / Process Queues

"As Process Executes, it changes States" → life cycle.
(Generation to Termination).

5 States

1. **New**: OS is about to Pick a Program & convert into Process → Process in the Stage of being Created.
2. **Ready**: The Process has all the resources it needs to run. Waiting to be assigned to the Processor
   *) Process in memory (Ready queue).
3. **Run**: Instructions are being Executed ; CPU is allocated.
4. **Waiting**: The Process is waiting for some I/o actions (or) Some events to occur. For eg → waiting for keyboard i/p, child-Process to finish, disk access request & so on...
5. **Terminated**: The Process has finished Execution. PCB entry removed from Process table.



*) Process Queues :-

a) Job Queue :- (from Disk → Memory).
- i) Processes in new State. (Disk to Ready)
- ii) Present in 2° memory.
- iii) Job Scheduler (Long Term Scheduler) (LTS) Picks Process from the pool (J.Q) and loads them into memory for Execution (R.Q).

*) LTS → It gives good blend of different cycle [I/O cycle & CPU cycle]

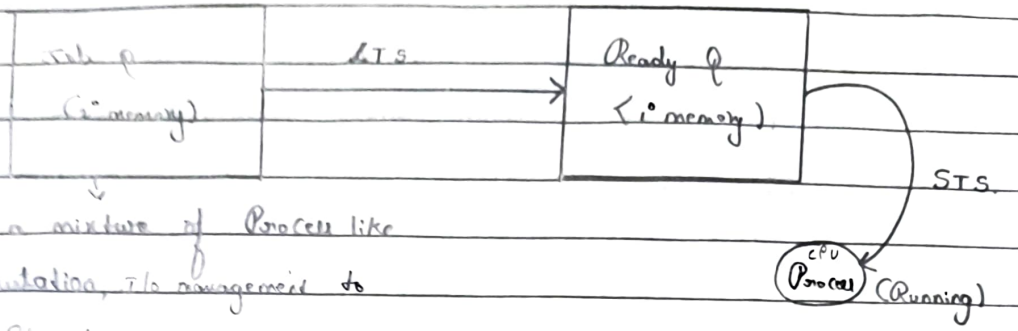
b) Ready Queue :-
- i) Process in Ready State (Ready to Running)
- ii) Present in main memory
- iii) CPU Scheduler (Short Term Scheduler) (STS) Picks Process from the R.Q and dispatch it to CPU. (depending on Priority).

* STS → has high frequency → No CPU idle time → Work very fast.

c) Waiting Queue :-
- i) Processes in Wait State.

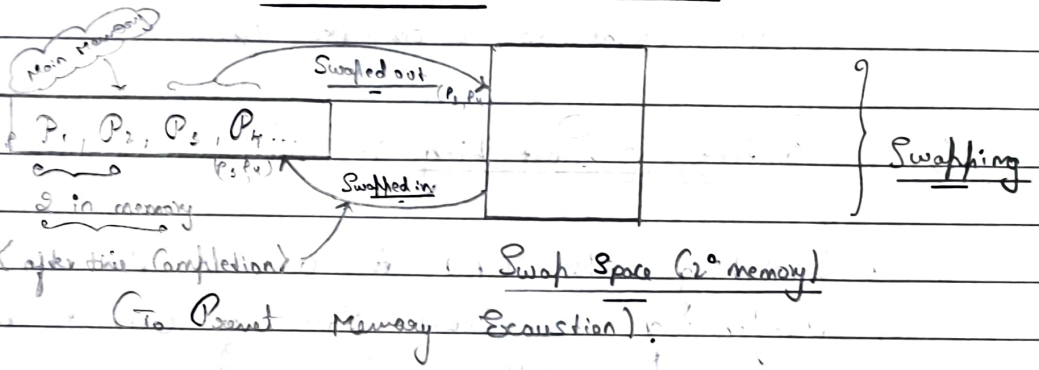
*) Degree of Multi-Programming :-
* Number of Processes in the memory.
* LTS Controls Degree of M.P.

*) Dispatcher :-
* The module of O.S that gives Control of CPU to a Process Selected by STS.



LTS → Select a mixture of Process like (I/o Computation, I/o management) to avoid Starvation.

*) MTS → Medium Term Scheduler :-

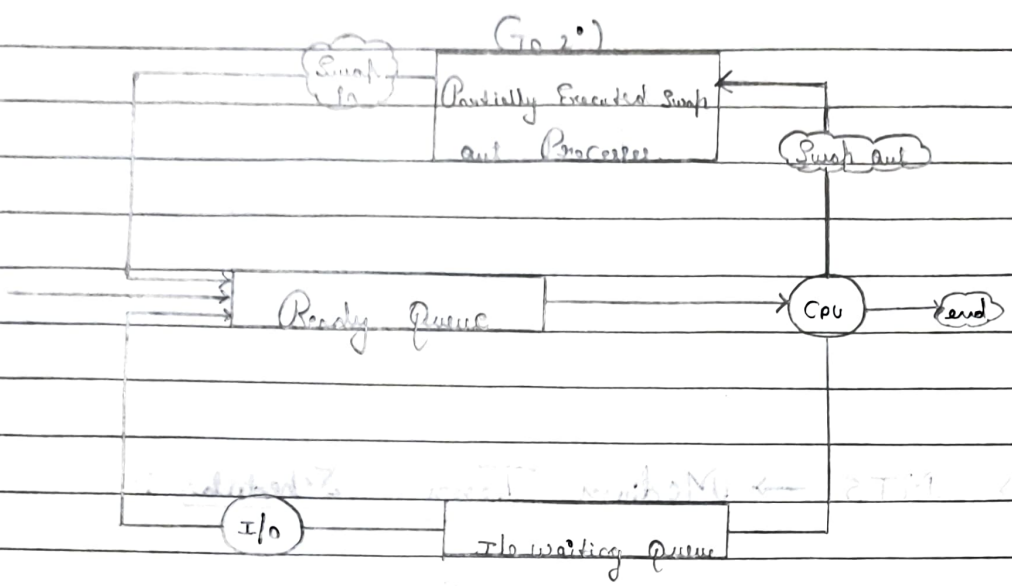


*) If Swapping :- (Main Context is with physical memory (RAM))

* Time Sharing System may have MTS
* It Remove Processes from memory to reduce degree of Multi-Programming.
* These removed Processes can be reintroduced into memory, and its Execution can be Continued where it is left off. This is called "Swapping".
* Swap-out and Swap-in is done by MTS.

↓

" Swapping is a mechanism in which a Process can be Swapped temporarily out of main memory (or move) to 2° Storage (disk) and make that memory available to other Processes. At Some time later, the System Swap back the Process from 2° Storage to main Memory ".

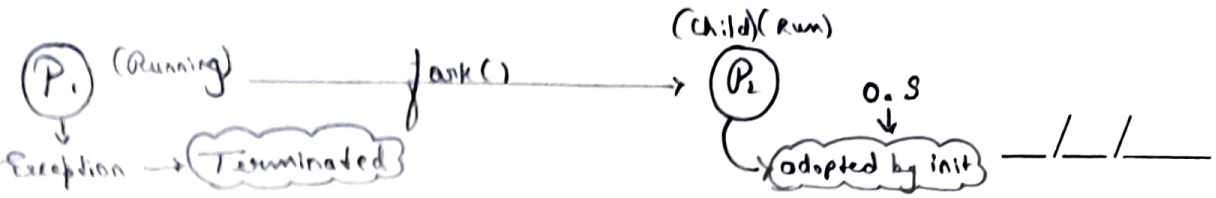


Context - Switching :- (Also called as with CPU time Sharing).

* Switching the CPU to another Process requires Performing a State Save of Current Process & a State restore of different Process.
* When this occurs, the kernel Saves the Context of old Process in its PCB & loads the Saved Context of new Process Scheduled to run.
* It is pure (Own Perspective) overhead, because System does no useful work While Switching (Ofc Ready & Process (user useful process is done during this)
* Speed varies from machine to machine, depending on memory Speed, no of registers that must be Copied etc.
* done by kernel. (Such a low-level work).

Orphan Process :-

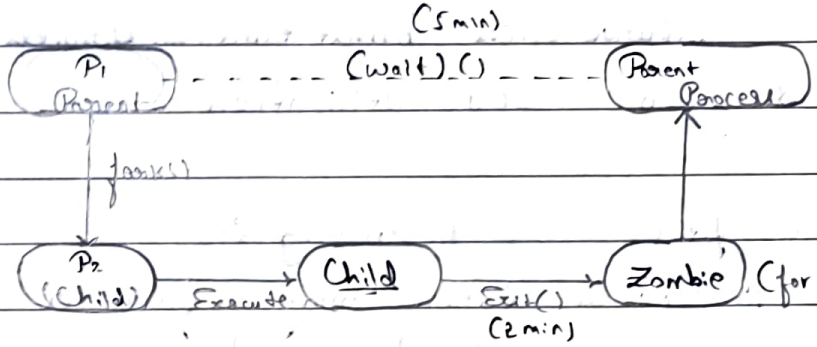
* The Process whose Parent Process has been terminated and it is Still running
* Orphan Processes are adopted by "init" Process.
* Init is the 1st Process of O.S (PID = 1).
* Process is done by fork() Command.



__/__/____

4. Zombie Process / Defunct Process :- (Z+).

* It is a Process whose Execution is Completed but it is Still having an entry in the Process table
* Zombie Process usually occur for Child Processes, as the Parent Process Still needs to read its Child's Exit Status. Once this is done using the wait System Call, the Zombie Process is eliminated from the Process table, this is known as 'Reaping Z.P'.



* Its entry in the Process table can only be removed, after the Parent Process reads the Exit Status of Child Process. Hence, the Childprocess remains a zombie till it is removed from Process table.