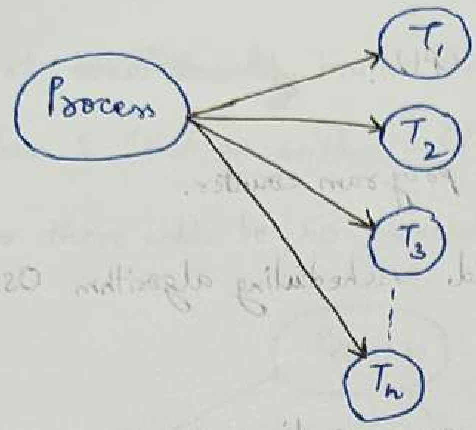
Concurrency

→ Concurrency is the execution of the multiple instruction sequences at the same time.

→ It happens in the OS when there are several process threads running in parallel.



Threads

{Recap}

Thread

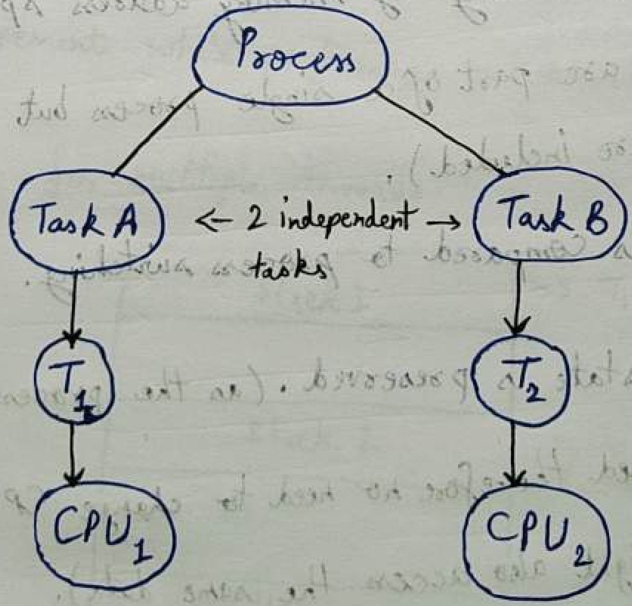
→ Single sequence stream within a process.

→ An independent path of execution in a process

→ Light-weight process.

→ Used to achieve parallelism by dividing a process's tasks which are independent path of execution.

eg:- Multiple tabs in a browser, text editor (when we are typing in an editor, spell checking, formatting of text and saving the text are done concurrently by multiple threads.)



How each thread get access to the CPU?

→ Each thread has its own program counter.

→ Depending upon the thread scheduling algorithm OS schedule these threads.

→ OS will fetch instructions corresponding to PC of that thread and execute instruction.

Thread Scheduling

→ Threads are scheduled for execution based on their priority.

→ Even though threads are executing within the runtime all threads are assigned processor time slices by the OS.

Thread context switching

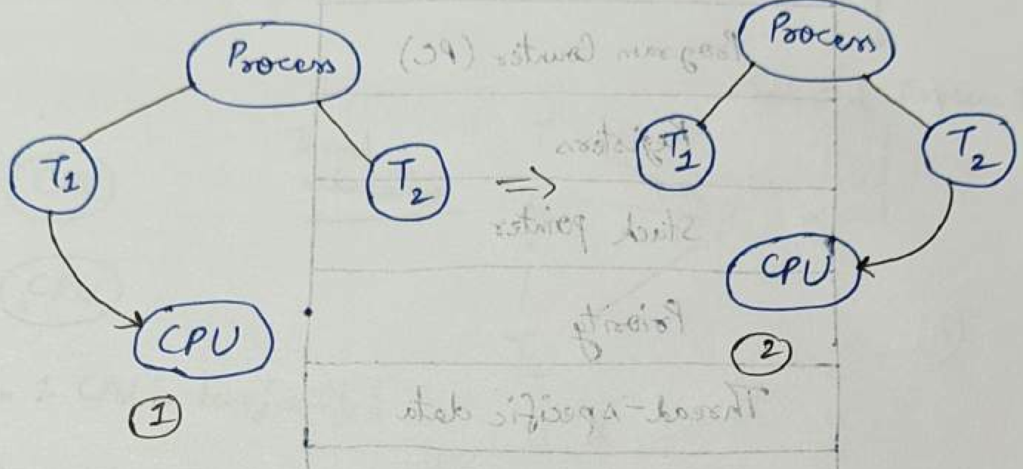
→ OS saves current state of thread and switches to another thread of same process.

→ Doesn't include switching of memory address space (as all the threads are part of a single process but program counter, registers & stack are included).

→ Fast switching as compared to process switching.

→ CPU's cache state is preserved. (as the process is same only thread is switched therefore no need to change CPU's cache state as other thread might also access the same data).

→ But ~~[illegible]~~ multi-threading is only useful in systems having more than 1 CPU as with 1 CPU the threads will be processed sequentially so there will be no parallelism.

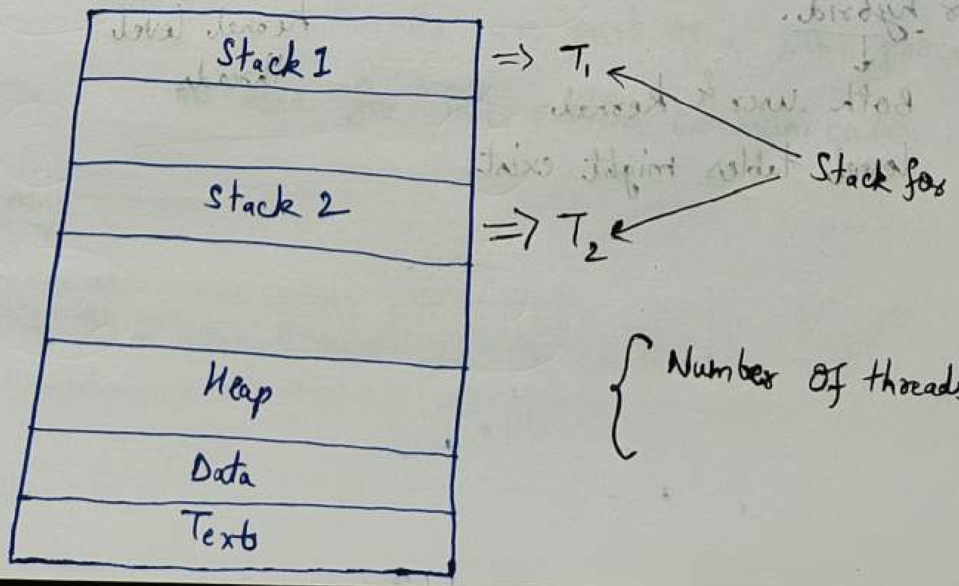


→ With 1 CPU first T_1 thread is executed in above diagram then T_2 thread is executed hence showing sequential processing. (Also 2 threads have to context switch for that single CPU).

→ I/O or time quantum ~~bas~~ based context switching is done here as well.

→ We have TCB (thread control block) like PCB for state storage management while performing context switching.

→ PCB for multiple threads:



{ Number of threads = Number of stacks }

→ TCB

Thread ID
Thread state
Program Counter (PC)
Registers
Stack pointer
Priority
Thread-specific data
Pointer to PCB

→ In multithreaded system there is a thread table also along with process table.

→ Thread table is used to manage threads.

→ A single process entry in the process table may point to multiple threads in the thread table.

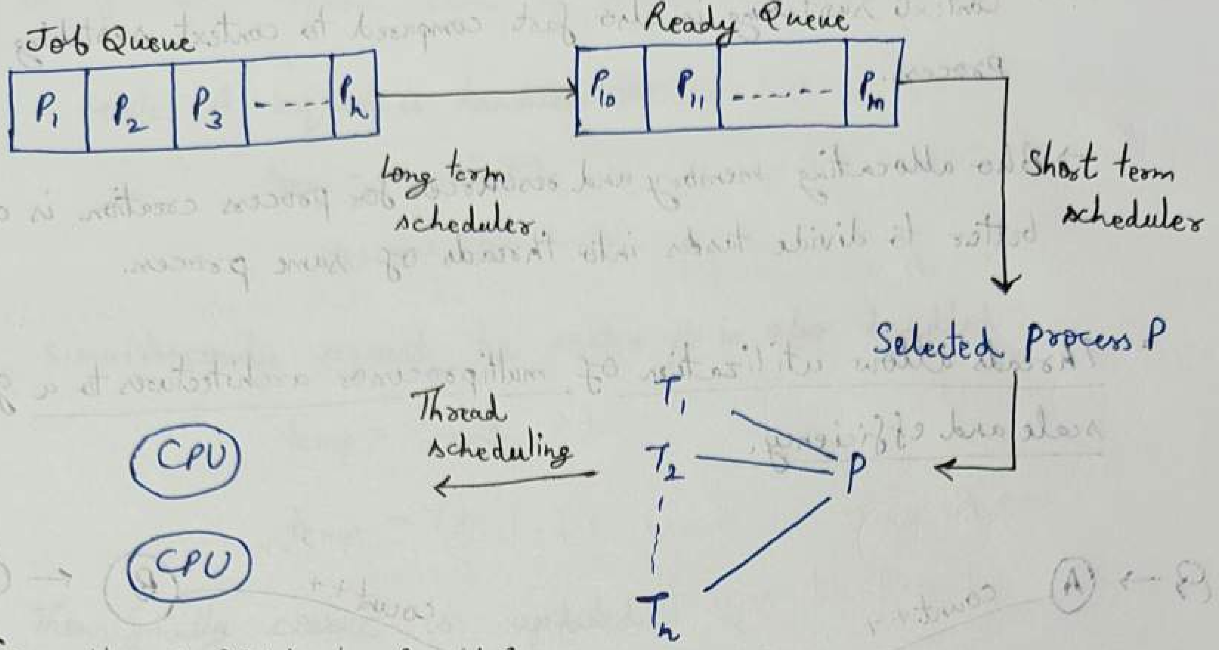
→ Process table is maintained ~~[illegible]~~ by OS

Thread table is maintained by either OS kernel or by thread library. or hybrid.

hybrid ↓ Both user & kernel level tables might exist.

OS kernel ↓ Kernel level threads

thread library ↓ User level threads.



{more than 1 CPU is beneficial}

Benefits of multi-threading.

- Responsiveness

  eg :- In MS Word text is formatted + it is taking input also so multiple tasks are being done at a time increasing responsiveness.

- Resource sharing : Efficient resource sharing

  → As all threads share the same resources like memory, etc.

  → As all threads are part of a same process therefore no need for inter process communication (IPC) also.

- Economy

  → It is more economical to create and context switch threads.

→ Context switching is also fast compared to context switching in process.

→ Also allocating memory and resources for process creation is costly so better to divide tasks into threads of same process.

6. Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.