Date: / /

Process Control Block:

Each process is represented in the OS^ by a PCB (also called task control block) as a Data structure

→ Process state –
Current state (The user wants to identify the state: new, ready, waiting)

→ Program number
Unique identification of process in order to track

→ Program counter
Address of next instruction to be executed for this process

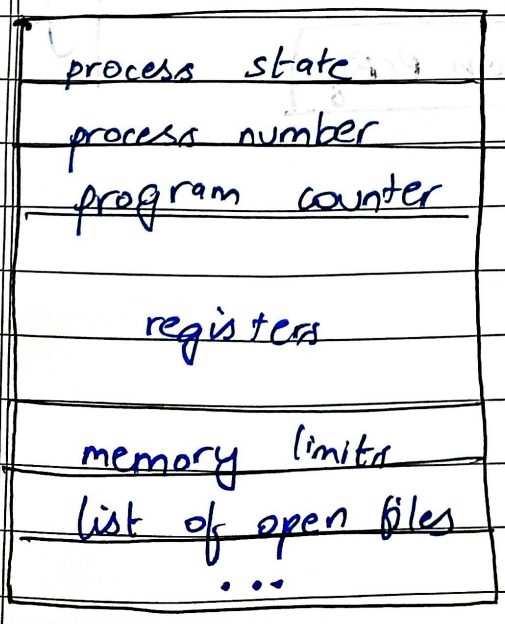
→ Pointer to parent process

→ Pointer to child process

→ CPU registers
particular register that are being used by a particular process

→ CPU scheduling information
Pointer to scheduling queue

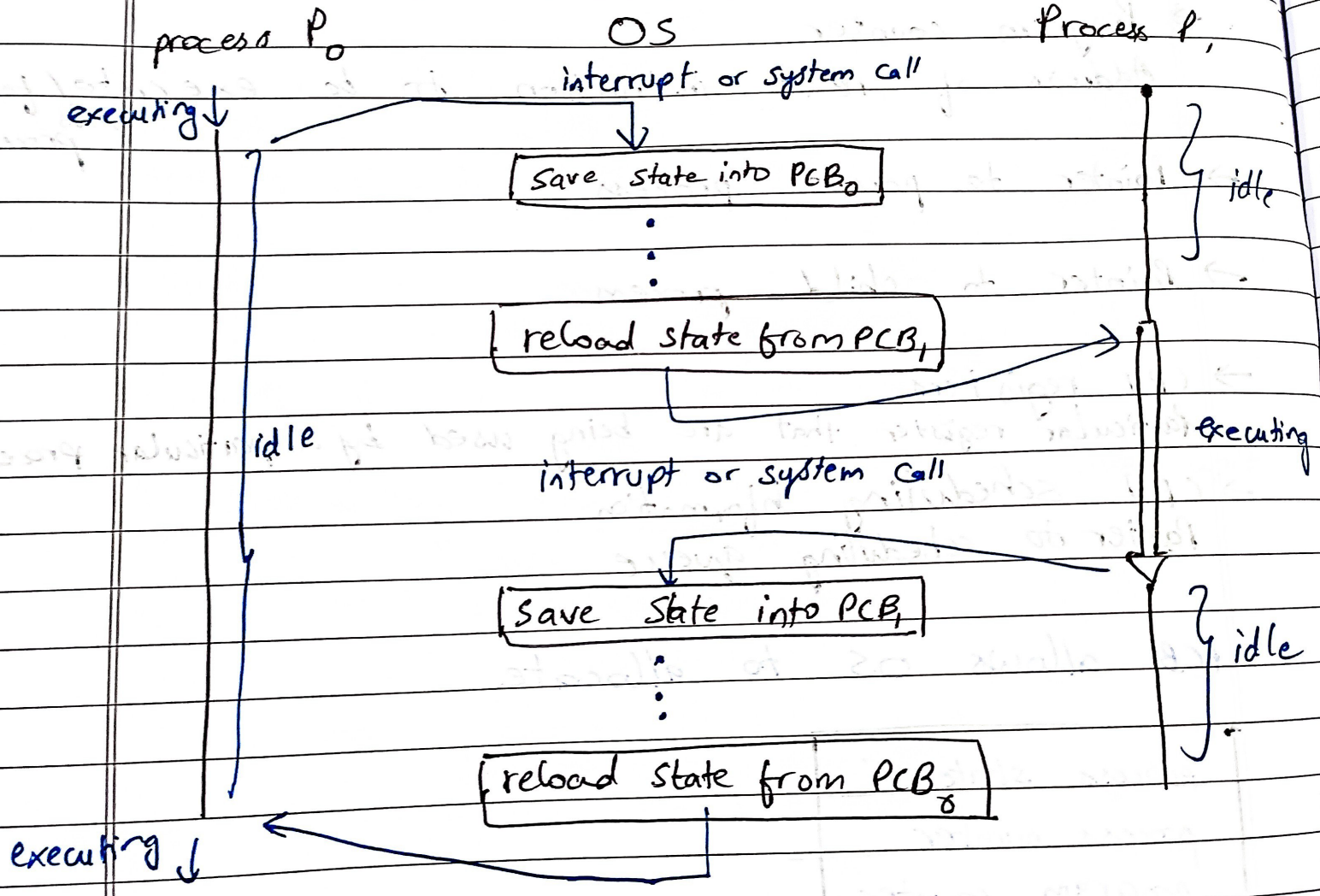
PCB allows OS to allocate



CPU Switch from Process to Process

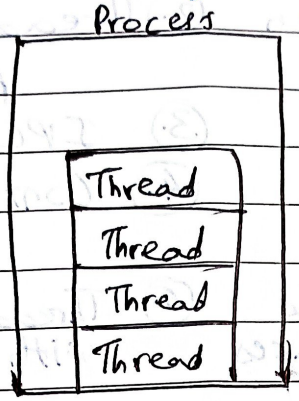
Consider P_1 and P_2, P_1 is in active state
P_2 is in ready state

Saving the load of old process and switching to the new process.

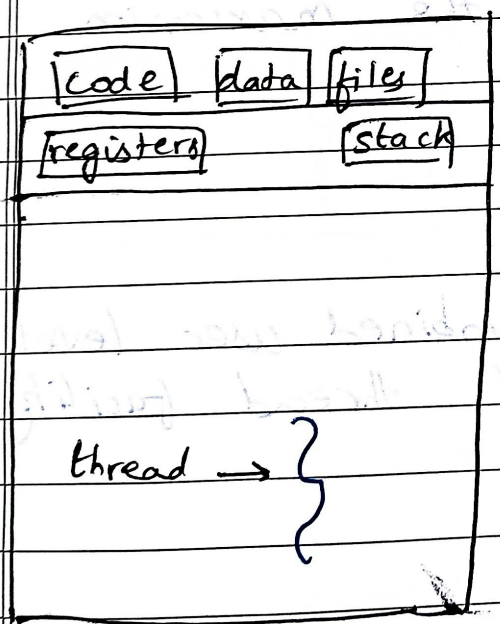


Thread

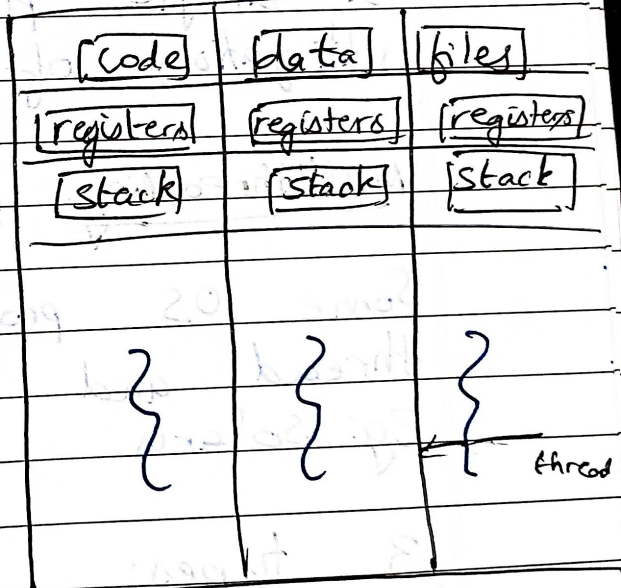
It is a unit of execution within a process.
A process can have from just one thread



A traditional process has a single thread of control.
If a process has multiple threads of control, it can perform more than one task at a time.



Single threaded process



Multi threaded process

Process	Threads
1. Processes are heavy weight operations	1. Threads are light weight operations
2. Every process has its own memory space	2. Threads use the memory of process they belong to
3. IPC is slow	3. IPC is fast
4. Context switching is more expensive	4. Context switching is less expensive
5. Processes don't share the memory with other processes	5. Threads share the memory with other threads of the same process

Multithreading

→ Execution of multiple threads at same time.
→ Allows multiple concurrent tasks to run within a single process for the maximum utilization of a CPU.

Multithreading Models

Some OS provide a combined user level thread and Kernel level thread facility.
Eg: Solaris

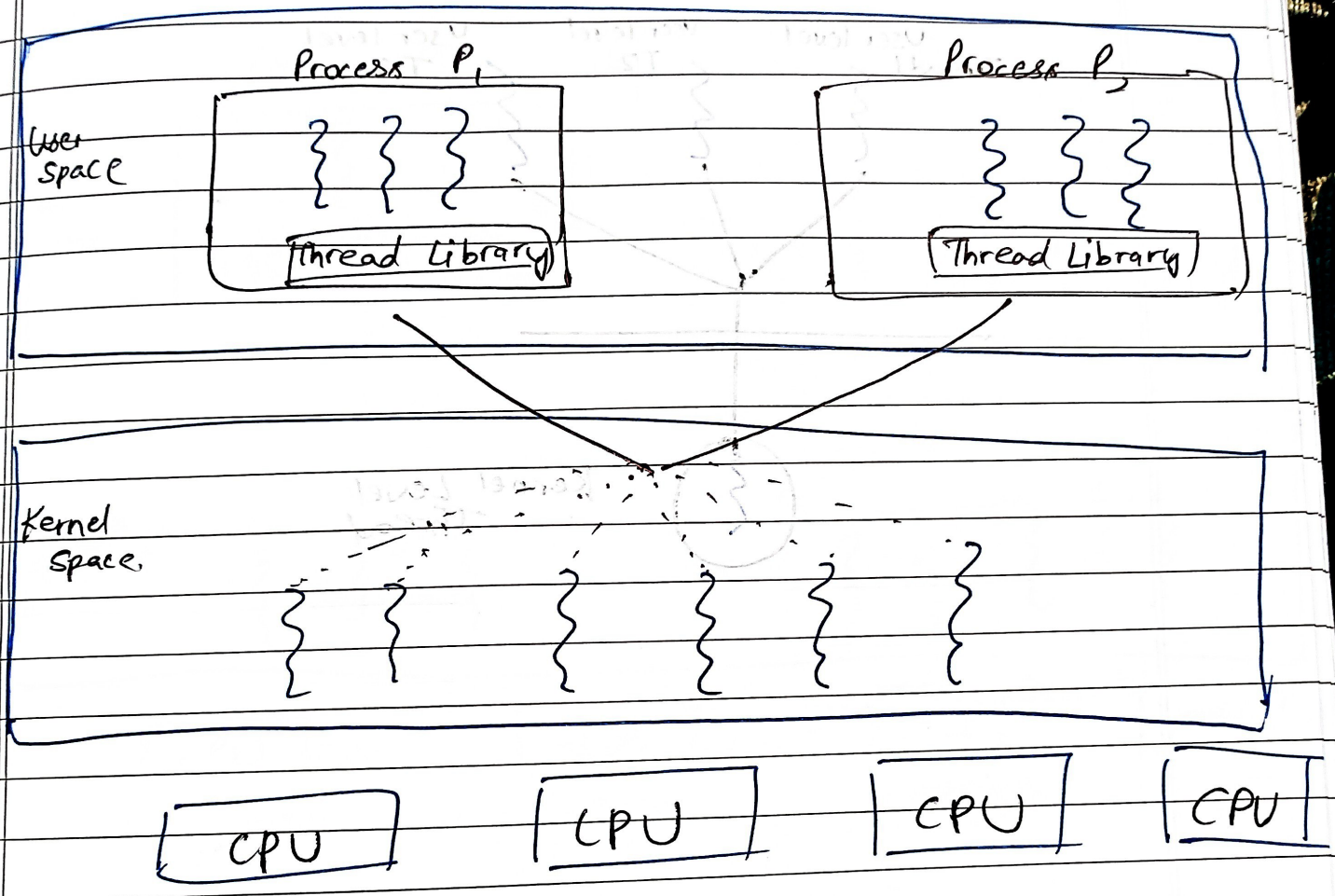
.3 types:
→ Many to many relationship
→ Many to one "
→ One to one "

Date: / /

Many to Many Model

→ Multiple user threads multiplex to same or lesser number of kernel level threads.

→ System doesn't block if a particular thread is blocked

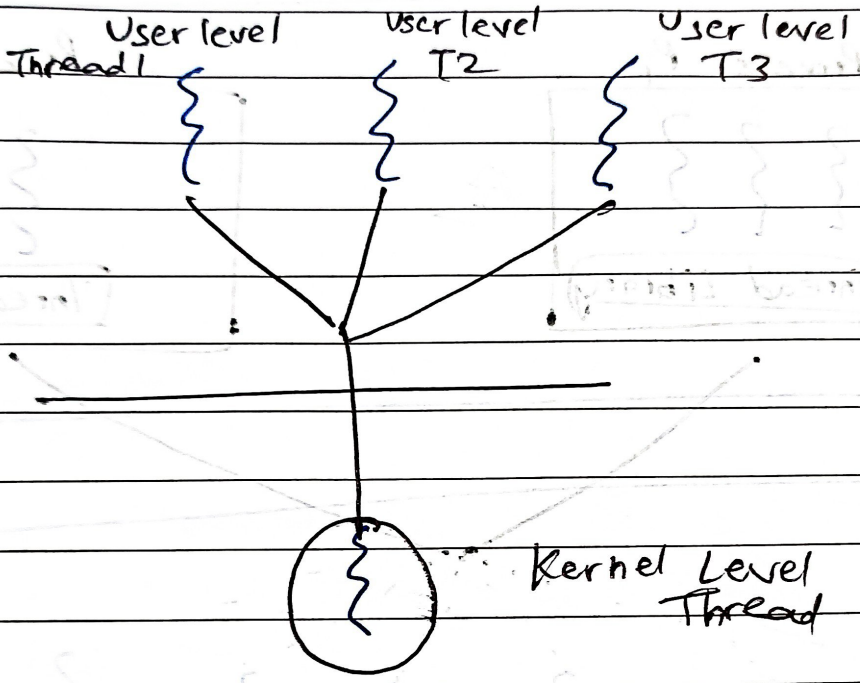


Date: / /

Many to One Model

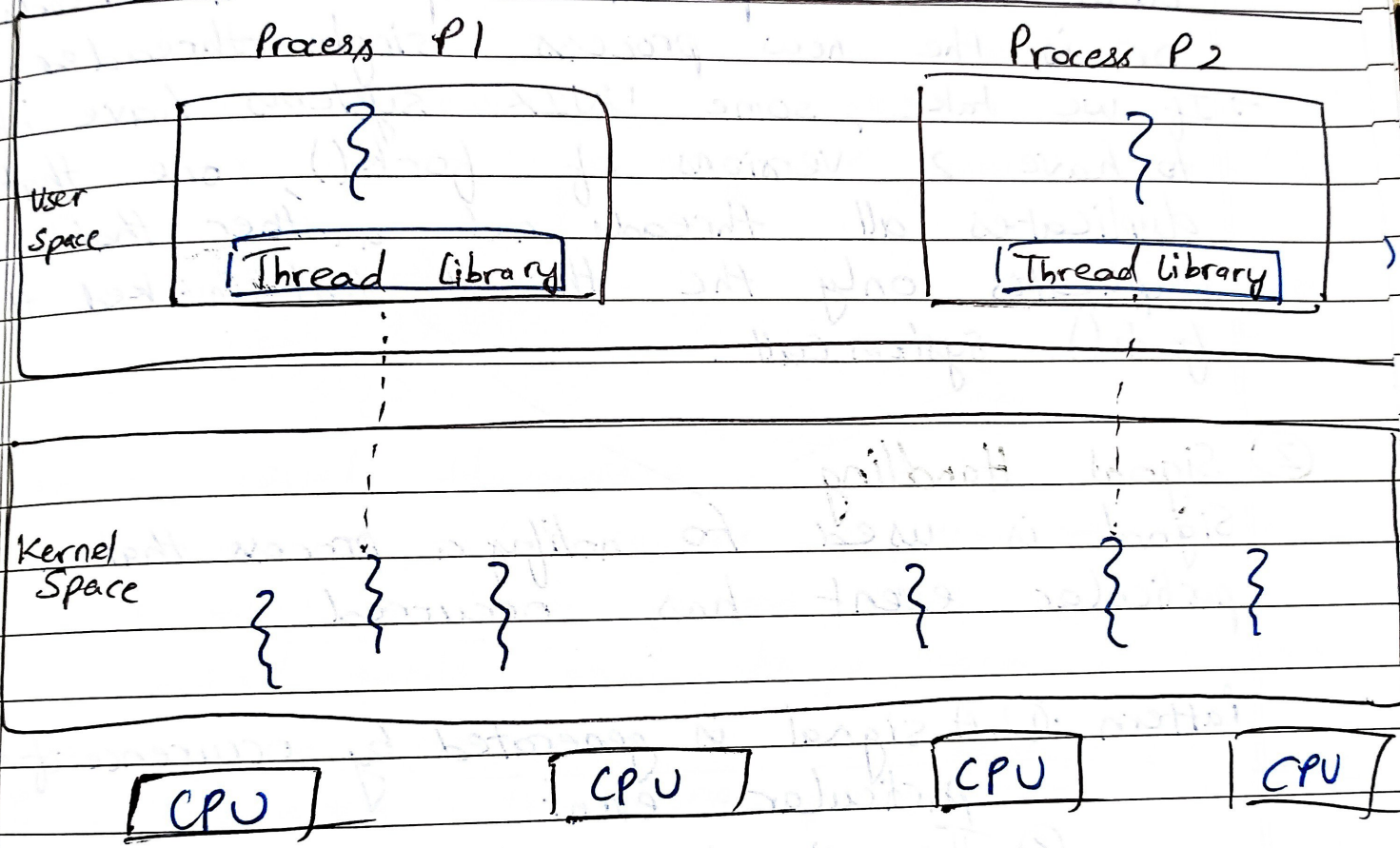
→ Maps many user level threads to one Kernel-level thread.

→ When thread makes a blocking system call the entire process will be blocked.



One to one Model

→ One-to-one relationship of user-level thread to the kernel-level thread.
→ Multiple threads to execute in parallel on microprocessor



Date: / /

Threading Issues

① The fork() and exec() system calls
The fork() is used to create a duplicate proces

If one thread in a program which calls fork(), does the new process duplicate all threads, or is the new process single-threaded?
→ If we take, some UNIX systems have chosen to have 2 versions of fork(), one that duplicates all threads and another that duplicates only the thread that invoked the fork() system call.

② Signal Handling
Signal is used to notify a process that a particular event has occurred.

Pattern: ① A signal is generated by occurence of a particular event
② The signal is delivered to a process
③ Once delivered, the signal must be handled

③ Thread Cancellation
Task of terminating a thread before it has completed.

Date: / /

Types of Threads:

(1) User Level Threads
User managed threads

Eg: POSIX, WIN 32, JAVA THREADS

Advantages
1. Doesn't need modification to OS
2. Simple representation (only PC, registers, stack)
3. Simple management
4. Fast and Efficient

(2) Kernel level threads
Kernel knows and manages the threads. Instead of thread table for each process, the kernel itself has thread table that keeps track of all the threads in system.
→ Also maintains traditional process table to keep track

Advantages
1. Scheduler can give more time to processes having more threads
2. Good for applications that frequently block

Disadvantages
1. Slow and inefficient

Eg: Windows XP, Solaris, Linux, Mac OS

Process Scheduling

Activity of the process manager that handles the removal of the running process from CPU and selection of another process.

When more than one process is runnable the OS must decide which one first. The path of the OS concerned with this decision is called scheduler and algorithm used is called scheduling algorithm

CPU scheduler
Select from among the processing memory that are ready to execute and allocates the CPU to one of them.

CPU scheduling decisions may take place when a process is switches from
1. running to waiting state or
2. running to ready state or
3. waiting to ready state

Once process is done it will terminate from the condition.

Dispatcher
Dispatcher module gives control of the CPU to the process selected by a short term scheduler

Date: / /

This involves 3 different categories:
① Switching context
② Switching to user mode
③ Jumping to proper location with user program to restart the program

After allocating the proper process to the CPU, then the scheduler criteria starts:

① CPU utilization
Keep the CPU as busy as possible (won't be idle)

② Throughput
Complete their execution per time unit

③ Turn around Time
~~Among~~ Amount of time to execute a particular process.

④ Waiting Time
Amount of time a process has been waiting in the ready queue

⑤ Response Time
Amount of time it takes from when a request was submitted until the response is produced

Date: / /

General goals of CPU scheduling:

1. Fairness → It is important under all circumstances to make sure that each process gets its fair share of CPU
2. Policy enforcement
3. Efficiency → Scheduler must keep the system busy
4. Response time → A scheduler must minimise the response time for interactive user
5. Turn-around time
6. Throughput

Two types of scheduling methods:

1. Preemptive scheduling:
If once a process has been given the CPU can taken away.
The strategy of allowing ~~running~~ processes that are logically runnable to be temporarily suspended is called preemptive scheduling.

2. Non-preemptive scheduling
A scheduling algorithm is called non-preemptive once a process has been given the CPU.
The CPU can't be taken away from the process

Date: / /

Scheduling Algorithms:

1. FCFS [First come first serve]
2. Round-Robin
3. SJF [Shortest Job first]
4. Priority scheduling
5. Multi-level queue scheduling
6. Multi-level feedback queue scheduling