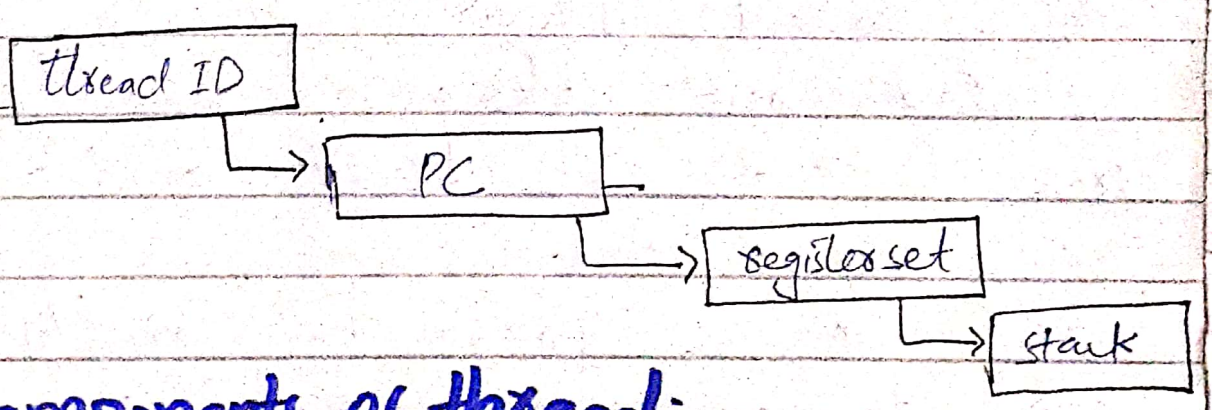
Thread

Ch#1

- A thread behaves like a process within a process but it does not have its own PCB.

What is thread? Also known as thread of execution or thread of control.

- A thread is a basic unit of CPU utilization. it is a sequence of instructions with in a process
- A thread is a light weight with limited no of instructions
- Thread address space is associated with process address space.
- A process can have several threads.
- A thread refers to a single sequential flow of activities being executed in a process
- A thread comprises:



Components of thread:

- Stack space
- Register set
- PC

⇒ A thread is also known as thread of execution or thread of control.

⇒ Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks.

⇒ The process can be split down into many threads.

Why we need thread?

⇒ To increase the performance of the applications.

⇒ Provide way to improve performance of the application through parallelism.

⇒ Context switching is faster.

⇒ Each thread belong to exactly one process and no thread exist outside a process.

⇒ When a process has multiple task to perform it independently we create thread.

⇒ To perform multitasking on single processor.

Advantages

★ Performance:
- Improve the overall performance, throughput, computational speed, responsiveness of an program

★ Concurrency:
- Provides concurrency within a process

★ Parallelism:
- parallel programming techniques are easier to implement

★ Reduced context switching time
- because virtual memory space remains the same

★ Utilization of multiprocessor architecture
- Multiprocessor architecture allow the reality of parallel processing.
- This is most efficient way of processing.
- This enable the utilization of the processor to a large extent & efficiency.

★ Resource sharing:

- Threads share the memory and resources of the process to which they belong by default. very easy & simply by using global variable

★ Economy:

- All threads in a process share the resources of that process so it is very cheaper to create and context switch the thread.

★ Responsiveness:

- more responsiveness in a multi-threading even if one thread is blocked or performing a lengthy process.
- Useful for designing interface.

Example of thread:

for example,

=> Multiple tabs in browser.

=> Useful for implementing webserver & network server.

Drawbacks:

- Blocking of parent thread will stop all child thread.
- Security.

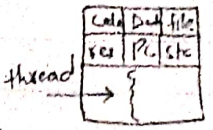
What is Single threading & multi-threading ??

Single threading

- Executing a single thread at a time is called single threading.

for example:

A simple calculator program that takes i/p from the user, performs calculations and display the result.



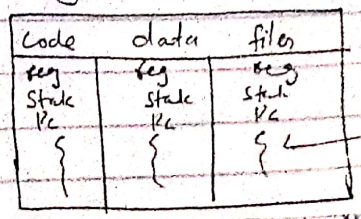
Multi-threading:

- The execution of more than one thread is called multithreading.

e.g.:- Most OS kernels are typically multithreaded.

Web browsers:-

- When we open a site one thread of web browser is working to get data from the server.
- Other thread is working to show image and UI to you.



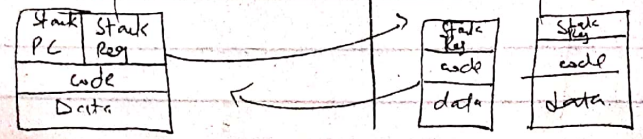
1 What is difference between Process & thread?

Process

- heavy weight process
- OS treats different process differently
- Different process have different copies of data, files, code
- Context switching is slower
- System calls are used to generate new process.
- Communication among processes occurs via OS & data copying.
- Independent
- Blocking a process will not block
- Expensive

Threads

- Light weight process.
- All user level threads treated as single task for OS.
- Share same code & data.
- Context switching is faster.
- There is no system call involved here.
- Communication among threads via memory
- Interdependent
- Blocking a thread will block entire process
- Inexpensive



Difference between multi-tasking & multi-threading?

Multi-tasking

Basics:

- The process of multi-tasking lets a CPU execute various tasks at the same time.

Working

- A user can easily perform various tasks simultaneously with their CPU using multi-tasking.

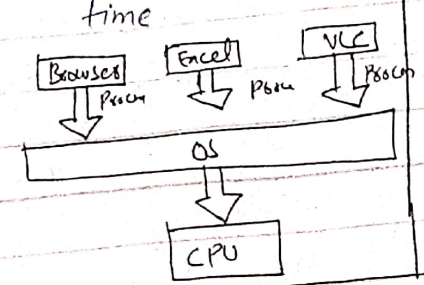
Resources & memory

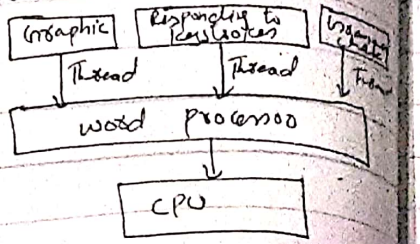
- The system needs to allocate separate resources & memory to different programs working simultaneously.

Multi-threading

- The process of multi-threading lets a CPU generate multiple threads out of a task & process all of them simultaneously
- A CPU gets to divide a single program into various threads so that it can work more efficiently & conviniently.
- System allocates a single memory to any given process in multi-threading.

Multiprocessing	- does not involve.
- involves multi processing among the various components.	
Speed of execution	- Comparatively faster.
- Slower	
Process Termination	o Takes up less time.
o Take more time	





Q: What is difference between Concurrency & parallelism?

Def (PP-2018/2019/17)

Concurrency:	Parallelism
Def:	
o A concurrent system supports more than one task by allowing all the tasks to make progress at the same time.	o A parallel system can perform more than one task simultaneously.
Achieve through:	- it is achieved by the use of several processors of CPU.
- Achieved via the interleaving of processes on the CPU.	
Control flow:	
o Non-deterministic	o deterministic
Processing units required	
o Single processing unit	o may not be achieved with a single processing unit.
Usage:	
o Deals with many things simultaneously.	o does multiple things simultaneously.
Make use of:	
o Use context switching	o uses multiple processors to run several processes.
Benefits:	
o increased the amount of work completed at a time	o improved throughput & computational

Examples:	
o Executes several apps simultaneously.	o Executes a web crawler on the cluster.

In Short:

o Concurrency may involve the various tasks executing and having overlapping time.

o Parallelism involves multiple tasks executing concurrently with the same start & finish time.

Q What is TCB? (PP-2014)

⇒ Stands for thread control block.

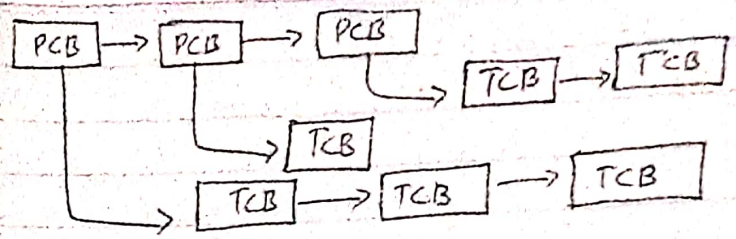
⇒ it is a data structure in the OS kernel which contains thread specific information needed to manage it.

⇒ The TCB is the manifestation of a thread in an OS.

⇒ Same like PCB.

Components:

- o Thread ID
- o Thread Priority
- Thread states
- o CPU info
- o Pointers



Q What is difference between TCB & PCB?

⇒ Both used in kernels, are data structures

⇒ TCB Contains info on a single thread:

o just processor state and pointer to corresponding PCB.

⇒ PCB contains info on the containing process.

o Address space & OS resources - but No processor state.

Long # Who manages thread?

Types of Threads.

(1) User level:

=> implemented by user and kernel is not aware of the existence of those threads.

=> Handles them as if they are single threaded process.

=> The user-level thread library includes the source code for thread creation, data transfer, thread destruction, merging, passing and thread scheduling.

Pros

- Fast (really light weight) (no system call to manage threads. The thread library does everything.)
- Can be implemented in an OS that does not support threading.
- Switching is fast. No switching from user to protected mode.
- Small, no kernel mode privileges required.

Cons:

- Scheduling can be issue (consider one thread that is blocked on an IO and another runnable)
- Lack of kernel & threads coordination (A process with 100 threads competes for a timeslice with a process having just 2 threads)
- Requires non-blocking system calls (if one thread invokes a system call, all threads need to wait)

Kernel-level-threads:-

- Are handled by the OS directly and all thread management is done by kernel
- These are slower than user-level threads
- Kernel maintains context information & perform thread creation, scheduling & management in kernel space.

Advantages :-

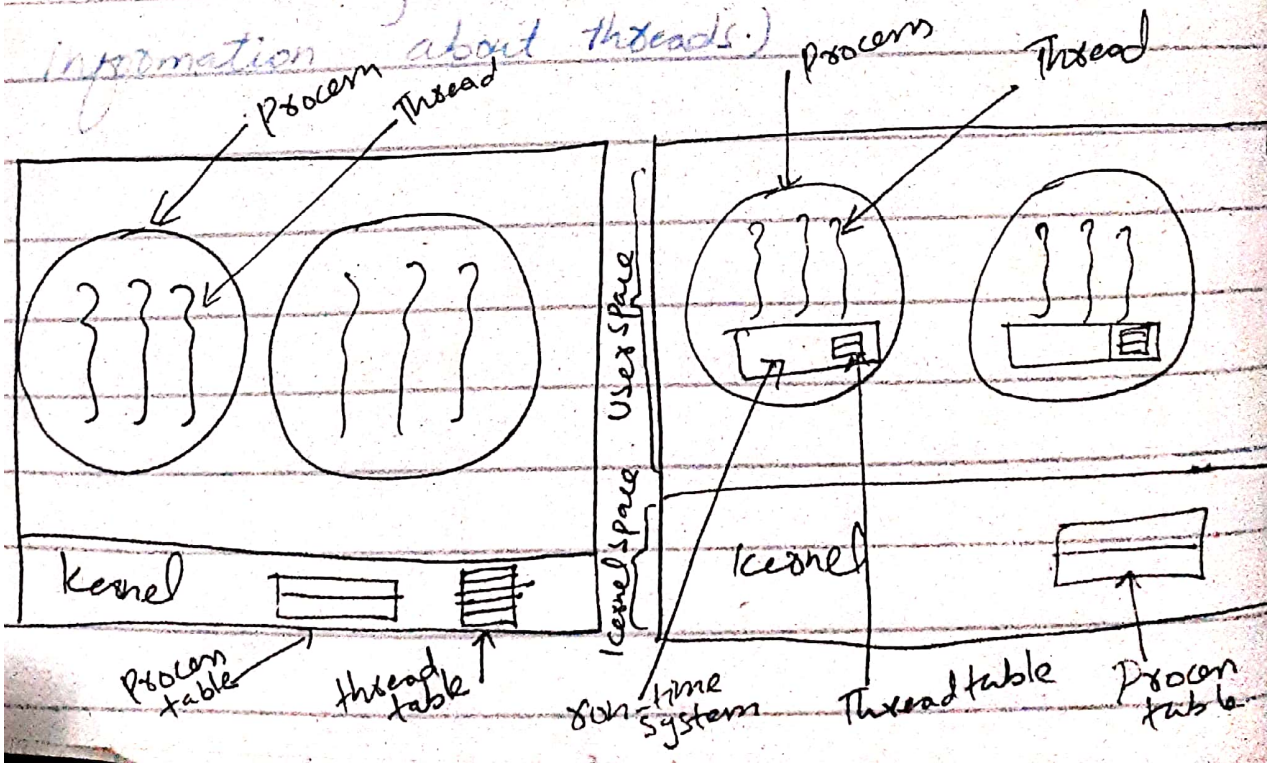
- Scheduler can decide to give more time to a process having large number of threads than process having small number of threads.
- Especially good for applications that frequently block.
- multiple threads of same process can be scheduled on different processors.

Cons:

=> Slow (they involve kernel invocations)

=> Specific to the OS.

=> Overheads in the kernel (since kernel must manage and schedule threads as well as processes. It require a full thread control block for each thread to maintain information about threads.)



User level thread VS Kernel-level thread

Implemented by:

- User
- OS

Recognize

- Does not recognize by OS.
- Recognize by the OS.

Implementation:

- Easy
- Complicated

Cotent switch time:

- Less
- more

Hardware support:

- No
- Yes

Blocking operation:

- Entire process will block if one user level threads performs blocking.
- if one kernel level thread perform blocking operation then another thread can continue execution.

Speed:

- Slow Creation & management.
- Quickly creation & management

OS:

- Any OS can support
- Any OS specific

Multithreading:	
o multi-thread App cannot take advantage of multi-processing	o kernels can be multi-threaded.
Example:	
o Java threads o POSIX threads	o window solaris o macos & Linux

#Multi-threading models:

=> A relationship must exist b/w user threads and kernel threads.

=> Three common ways to establish relationship:

- o M-2-1
- o 1-2-1
- o M-2-M

M-2-1

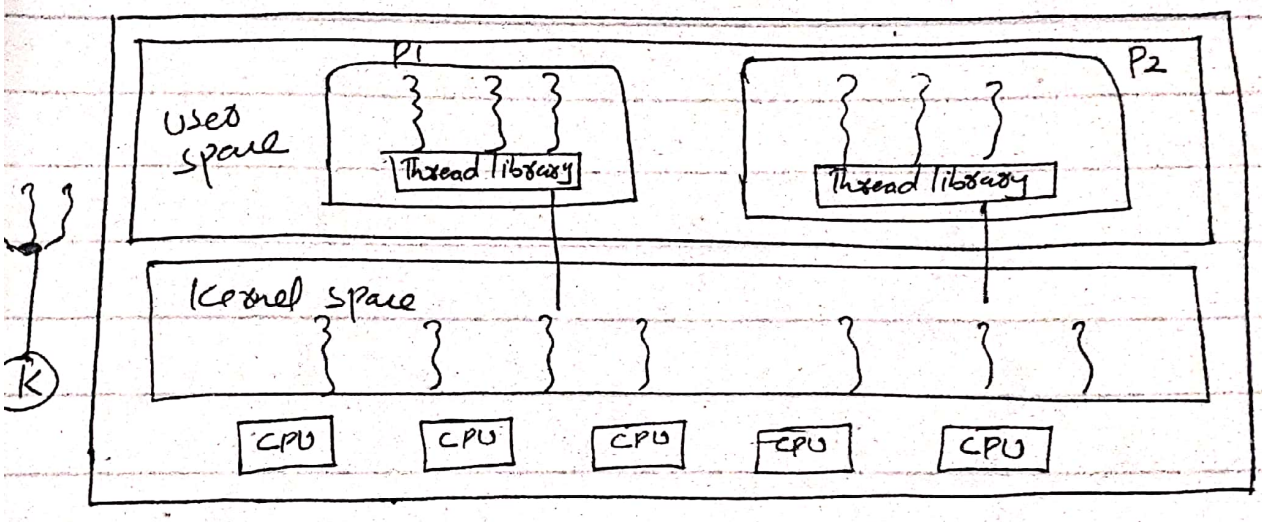
- o Maps many user-level threads to one kernel thread.
- o Thread management is done by thread library.

Pros:

- Efficient because management is done by the thread library in user space.

Cons:

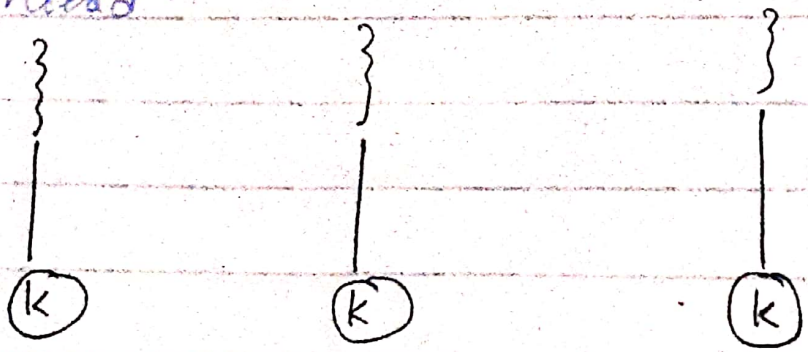
- Blocking
- No parallel, Thread can enters the kernel, one thread at a time enters the kernel.



Use: Used by windows, green threads in Solaris.

1-2-1 :-

- Maps each user thread to a kernel thread.



Pros:

- Provides concurrency
- no blocking
- Efficiency

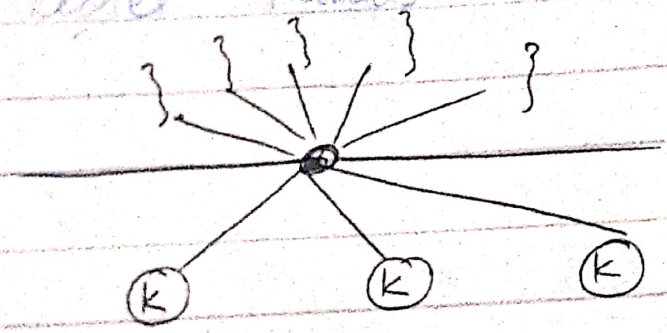
Cons:

- Performance decreases
- Speed slow

Use: Linux, windows 2000, Window XP

M-2-M :-

- Multiplexes many user level threads to a smaller or equal number of kernel threads.



-> > greater

= < less

Pros:

- Parallel
- efficiency
- No - block
- flexible

Use:

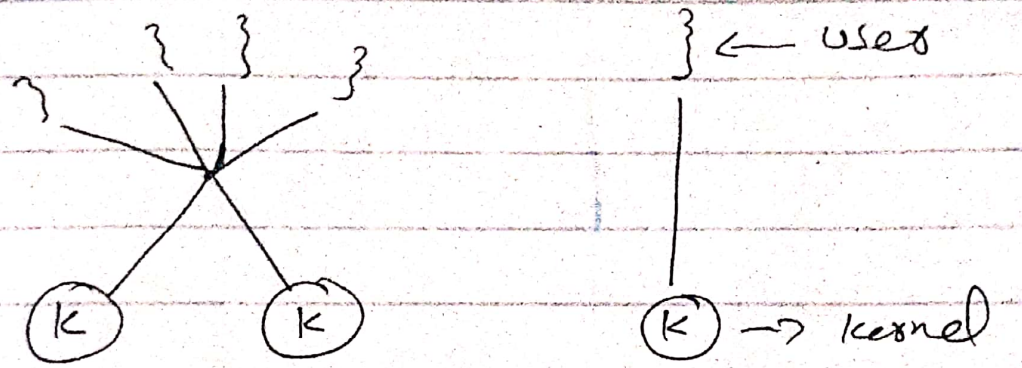
- Solaris, HP UX

Two - level - model:

=> Variation of m-2-m

=> Advantages of same like m:2-m model.

=> one thing is, User level thread that need to be done faster or require more time for that specific user level thread we make separate kernel level thread.



=> It is combination of m-2-m & 1-2-1.

What is thread library?

=> Provides an API to create & manage threads.

Two ways implementing thread:

1) Provide a library in user space with no kernel support.

- function call is local - No system call
- All code & data structures exist in user

(2) To implement a kernel level thread supported by the OS:

Types:

- POSIX (user + kernel side)
- PThreads:
- window & java (JVM → OS → kernel)
- Win32 - (kernel)

Window XP	Linux
Win32 • EThread → purely kernel • KThread • TEV → user Process ss/ thread	• fork() • clone() • Process / thread

What are thread cancellation approach?

⇒ Two approaches:

- Asynchronous
- Synchronous

What are thread issues?

While implementing multi-threading various issues we have created.

) fork() & exec() system call

) Thread Cancellation.

3) Signal handling of Thread
4) Thread pools
Specific data

Q# PP- Question:

=> Content switching b/w kernel threads typically requires saving the value of the CPU registers from the thread being switched out and restoring the CPU registers of the new thread being scheduled.

=> Content of a thread is represented in the TCB.

=> Content switch is occurs when a single CPU resource is share among several processes.

=> In short, saving the state of the old thread and loading the saved state for the new process.

Ch#4 Short Questions (PP-2014-2020)

- Difference b/w concurrency & parallelism
- What are types of threads? (Repeat)
- Write down the name of primary thread libraries?
- What is multi-threading? (Repeat)
- Difference b/w user-level threads & kernel level threads?
- What is TCB?
- What are threads?
- Pros & cons of user level threads.

Long

What are two differences b/w kernel level & user level threads?
Explain different multi-threading models? (Repeat)
Describe the actions taken by a kernel to context switch b/w kernel level threads?
Describe the threading issues?

(Ch#4) Practice exercise

4.1) Provide two programming examples in which multi-threading provides better performance over single-thread

→ The process of executing multiple threads simultaneously is known as multi-threading

e.g
- Matrix multiplication
- UI updates
- web servers

4.4) Under what circumstances user level thread are better?

- Easier and faster to Create.
- more easily managed.
- Run on any OS.
- No Kernel privileges required for thread switching.

1.5) Describe the actions taken by a kernel to context-switch b/w kernel threads; (PP-2022)

- Saving the rest of registers, as well as other machine state, such as the state of the floating point registers, in the PCB is done by the Clock interrupt handler.
-

A context switch occurs when the kernel transfers control of CPU from an executing process to another that is ready to run.

- The kernel saves the state of the current thread, and then restores the state of the new thread being scheduled.

4.6) What resources are used when a thread is created? How do they differ from those used when a process is created?

=> When a thread is created fewer resources are used:

- PC
- Stack
- registers

=> While creating a process requires allocating a PCB, a rather large data structures.

4.13) Is it possible to have concurrency but not parallelism?

- Yes, it is possible.
- Concurrency means where two different threads start working together in an overlapped time period, it does not mean they run at the same instant.