What is a thread?:

→ Process has resource ownership & scheduling as it's characteristics

→ The unit of dispatching is thread (Lightweight Process)

→ Path of execution of instruction

→ Unit of resource ownership is task

→ The ability of an OS to support multiple paths in a single process

Server program handling multiple clients at the same time

→ Before threads, there was 1 process for each client

→ Now each client has 1 thread

Another

→ spell check in

Example

Word Program

→ 1st Process → Inp → Out
2nd Process → Spellcheck

Single Thread Approaches

→ MS-DOS & Unix

Multi thread Approaches

→ Java runtime is single process with multiple threads

→ Windows → multiple thread Multi thread

Process

→ Unit of resource allocation + protection

→ Virtual address space of process

→ Protected access to
① Processor
② Other Proc
③ Files

④ I/O

One or more threads in processes → Execution, local variables

→ Thread is like an independent program counter within a process.

Threads vs processes → Kernal stack & User stack with thread control block

File Server Application → Multi client requests

Threads use in a single user system → Foreground/Background

→ Async processes (Backup in Background)

↳ Independent of what else you are doing

→ Organisation

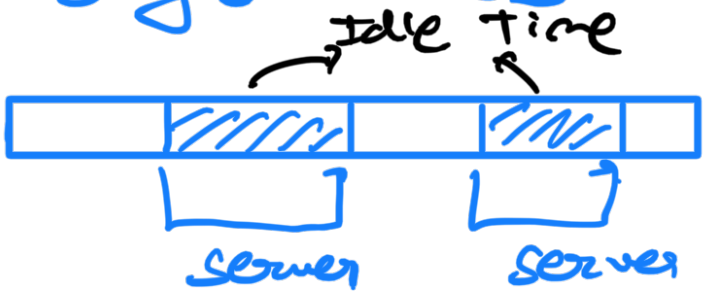
Management at process level → Suspension / Termination of threads vrt processes

States of a Thread → Running, Ready & Blocked

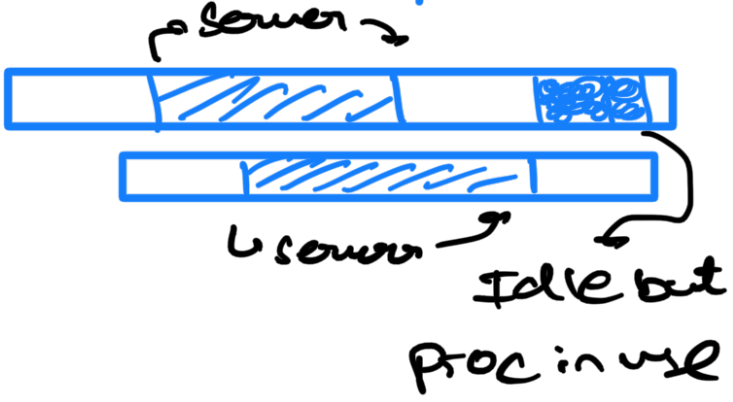
→ No sense to suspend

→ If a process is swapped out, all threads are swapped out.

Remote Procedure Call → Single Process



Multithreading → One thread per server



Thread Synchronisation → Two threads trying to add element to

a doubly linked list (element last)

Categories of Threads → User vs Kernal level

User level Threads → All management by application, no role of Kernal

→ Kernal is not aware of threads; it sees as a single process.

Relationship b/w Thread & Process → If a thread wants to open a file; it is requesting a system call
↳ Thread is running but the whole process is blocked

→ Thread 2 is blocked; waiting for event

from thread 1.

Advantages

→ Scheduling can be application specific
→ Thread scheduling can be customised
→ Can run on any OS ; as it is handled by the library.

Disadvantages

→ One thread can block the whole process
→ NO advantage of multiprocessing (as Kernal is unaware)

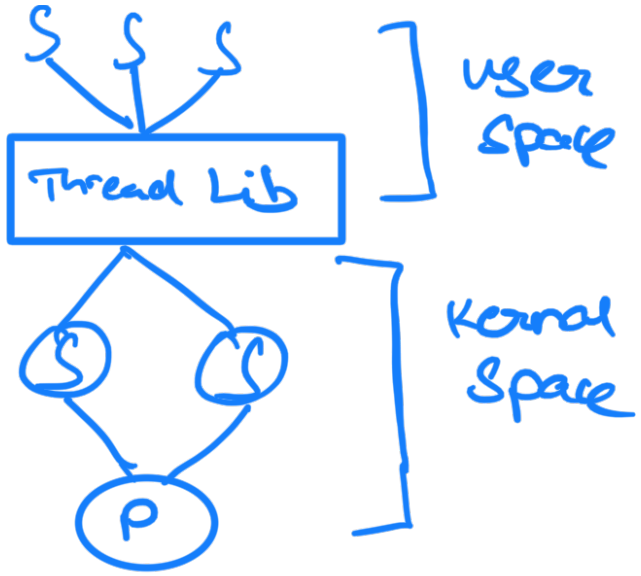
Kernal level Threads
Kernal is aware

→ Transfer of control from one thread to another → Requires mode switching

Time Comparison

→ ULT < KLT < Process

Combined Approach



→ 1:M is distributed (cloud systems)

Thread and Processes

Table 4.2 Relationship Between Threads and Processes

Threads:Processes	Description	Example Systems
1:1	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementations
M:1	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Solaris, Linux, OS/2, OS/390, MACH
1:M	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
M:N	Combines attributes of M:1 and 1:M cases.	TRIX

→ m to 1 is the most common

Summary: Threads - Single vs multi Threads - Why to use threads - Comparison with a process - Management - States of a thread - Categories - User vs Kernal Level - Advantages Disadvantages of each - Combined approach - Diff types of Relationships.