Great Place To Work. Certified FEB 2023 – FEB 2024 INDIA

UPES

UNIVERSITY OF TOMORROW

nirf RANKING 2023

#RANKED 52 IN INDIA

#University Category

QS WORLD UNIVERSITY RANKINGS

NO.1 PVT. UNIVERSITY IN ACADEMIC REPUTATION IN INDIA

ACCREDITED WITH GRADE A NAAC

ACCREDITED **GRADE 'A'** BY NAAC

E-LEAD

PERFECT SCORE OF **150/150** AS A TESTAMENT TO EXECEPTIONAL E-LEARNING METHODS

Unit 2 : Process and Thread Management

Lecture 6

Submitted by:

Dr Khushboo Jain

School of Computer Science

UPES, Dehradun

India

UPES
UNIVERSITY OF TOMORROW

Table of Contents

UPES
UNIVERSITY OF TOMORROW

Learning & Course Outcomes

LO1: To understand Threads and Their Benefits
LO2: To evaluate Multi-threading Models
LO3:To differentiate Types of Threads

CO2: Evaluate and analyze process and thread scheduling techniques, discerning their benefits and challenges.

UPES
UNIVERSITY OF TOMORROW

Motivation

- Most modern applications are multithreaded
- Threads run within application
- Multiple tasks with the application can be implemented by separate threads
  - Update display
  - Fetch data
  - Spell checking
  - Answer a network request
- Process creation is heavy-weight while thread creation is light-weight
- Can simplify code, increase efficiency
- Kernels are generally multithreaded

UPES
UNIVERSITY OF TOMORROW

Multithreaded Server Architecture

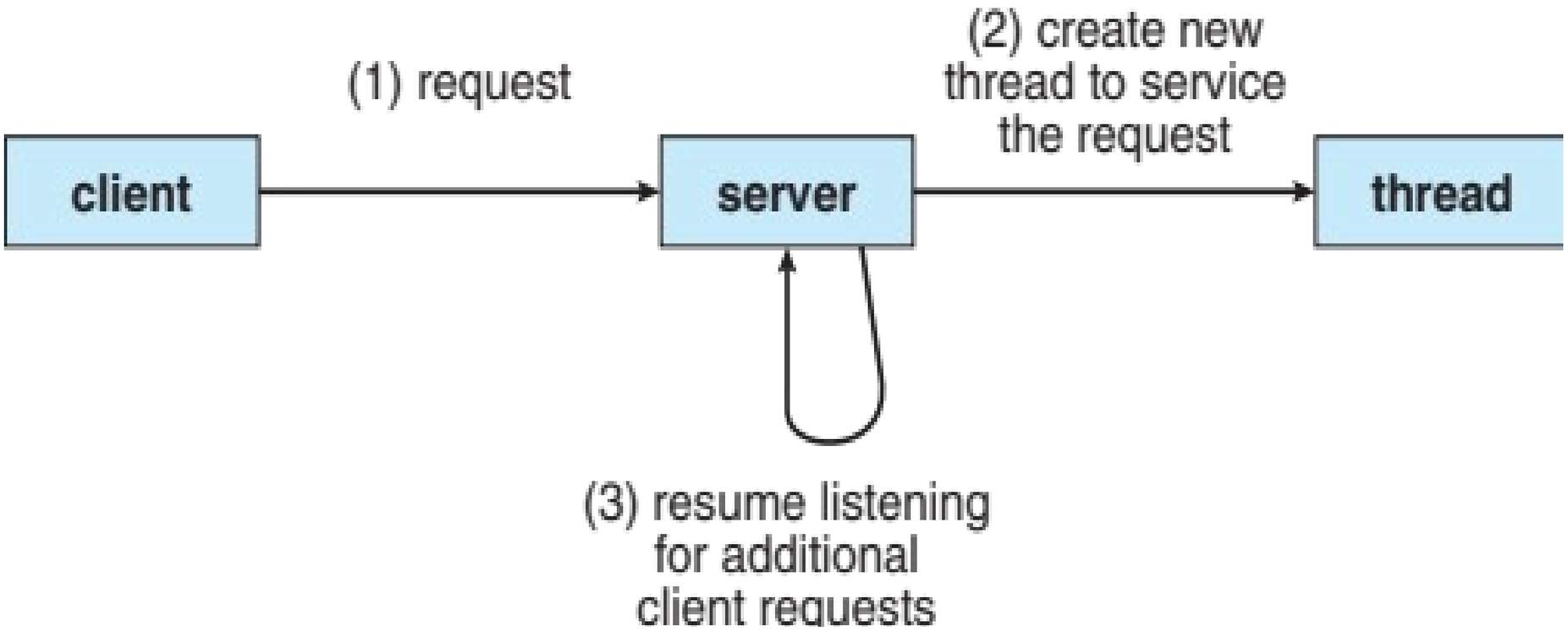

Fig 1. Multithreaded Server Architecture [1]

Benefits

- **Responsiveness –** may allow continued execution if part of process is blocked, especially important for user interfaces
- **Resource Sharing –** threads share resources of process, easier than shared memory or message passing
- **Economy –** cheaper than process creation, thread switching lower overhead than context switching
- **Scalability –** process can take advantage of multiprocessor architectures

Multicore Programming

- **Multicore or multiprocessor** systems putting pressure on programmers, challenges include:
  - **Dividing activities**
  - **Balance**
  - **Data splitting**
  - **Data dependency**
  - **Testing and debugging**
- **Parallelism** implies a system can perform more than one task simultaneously
- **Concurrency** supports more than one task making progress
  - Single processor / core, scheduler providing concurrency

Multicore Programming (Cont.)

- **Types of parallelism**
  - D**ata parallelism** – distributes subsets of the same data across multiple cores, same operation on each
  - **Task parallelism** – distributing threads across cores, each thread performing unique operation
- As # of threads grows, so does architectural support for threading
  - CPUs have cores as well as **hardware threads**
  - Consider Oracle SPARC T4 with 8 cores, and 8 hardware threads per core

Concurrency vs. Parallelism

Concurrent execution on single-core system:

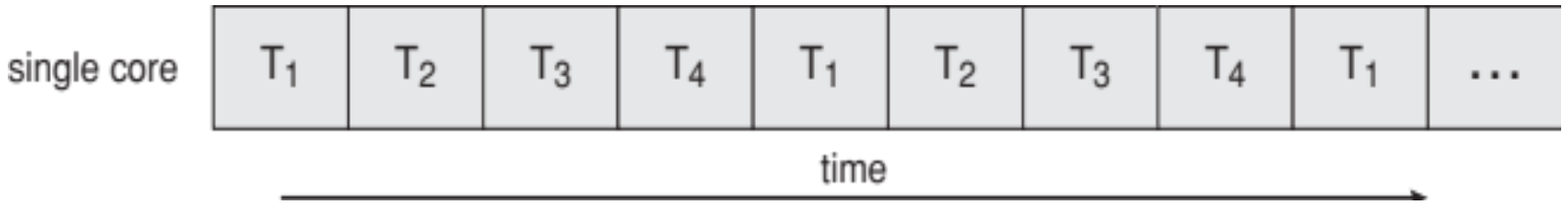

Fig 2. Concurrency [1]

Parallelism on a multi-core system:

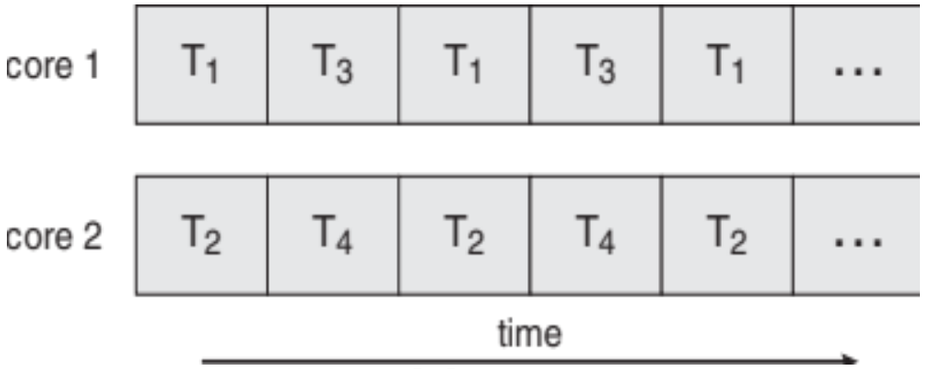

Fig 3. Parallelism [1]

Single and Multithreaded Processes

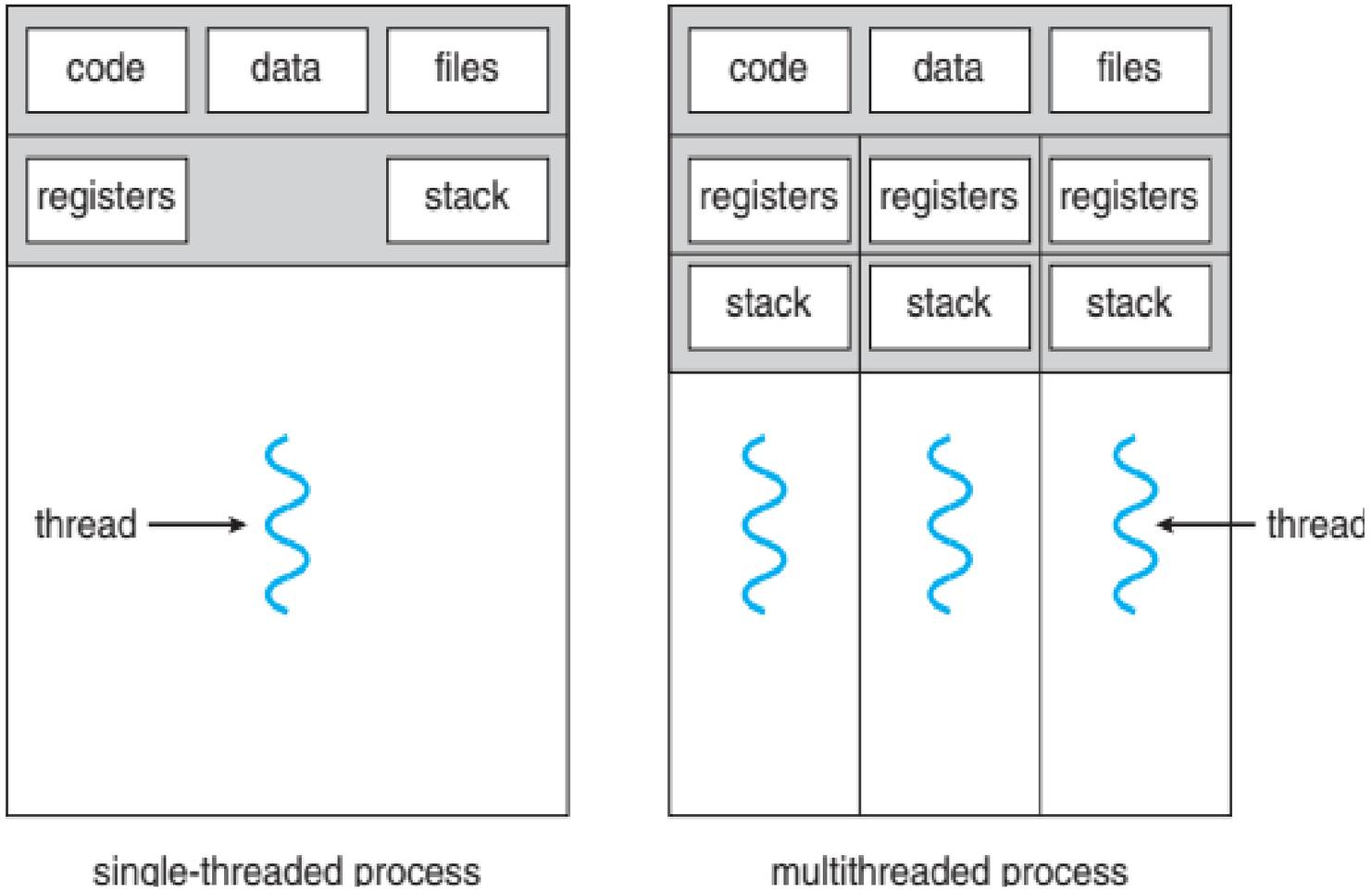

Fig 4. Single and Multithreaded Processes [1]

Concurrency vs. Parallelism

- Identifies performance gains from adding additional cores to an application that has both serial and parallel components
  - S is serial portion
  - N processing cores

$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

- That is, if application is 75% parallel / 25% serial, moving from 1 to 2 cores results in speedup of 1.6 times
- As N approaches infinity, speedup approaches 1 / S
- **Serial portion of an application has disproportionate effect on performance gained by adding additional cores**

UPES
UNIVERSITY OF TOMORROW

Numerical of Amdahl's Lab

Exercise and solution on Amdahl's Law

1. What is the overall speedup if you make 10% of a program 90 times faster?
2. What is the overall speedup if you make 90% of a program 10 times faster?

- Amdahl's law

$$OverallSpeedup = \frac{1}{(1-f)+\frac{f}{s}}$$

- What is the overall speedup if you make 10% of a program 90 times faster?

$$\frac{1}{(1-0.1)+\frac{0.1}{90}} \approx \frac{1}{0.9011} \approx 1.11$$

- What is the overall speedup if you make 90% of a program 10 times faster

$$\frac{1}{(1-0.9)+\frac{0.9}{10}} = \frac{1}{0.19} \approx 5.26$$

User Threads, Kernel Threads and Hybrid Threads

- **User threads -** management done by user-level threads library
- Three primary thread libraries:
  - POSIX **Pthreads**
  - Windows threads
  - Java threads
- **Kernel threads** - Supported by the Kernel
- Examples – virtually all general-purpose operating systems, including:
  - Windows
  - Solaris
  - Linux
  - Tru64 UNIX
  - Mac OS X
- **Hybrid thread –** Combination of both

Multithreading Models

- Many-to-One
- One-to-One
- Many-to-Many

Many-to-One Model

- Many user-level threads mapped to single kernel thread
- One thread blocking causes all to block
- Multiple threads may not run in parallel on muticore system because only one may be in kernel at a time
- Few systems currently use this model
- Examples:
  - Solaris Green Threads
  - GNU Portable Threads

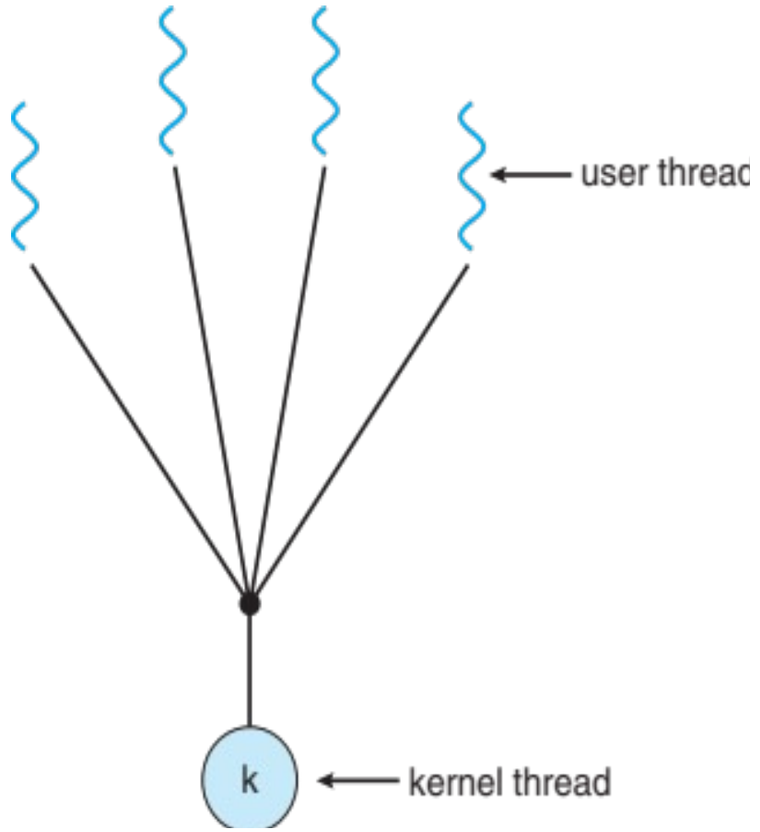

Fig 5. Many-to-One Model [1]

One-to-One Model

- Each user-level thread maps to kernel thread
- Creating a user-level thread creates a kernel thread
- More concurrency than many-to-one
- Number of threads per process sometimes restricted due to overhead
- Examples
  - Windows
  - Linux
  - Solaris 9 and later

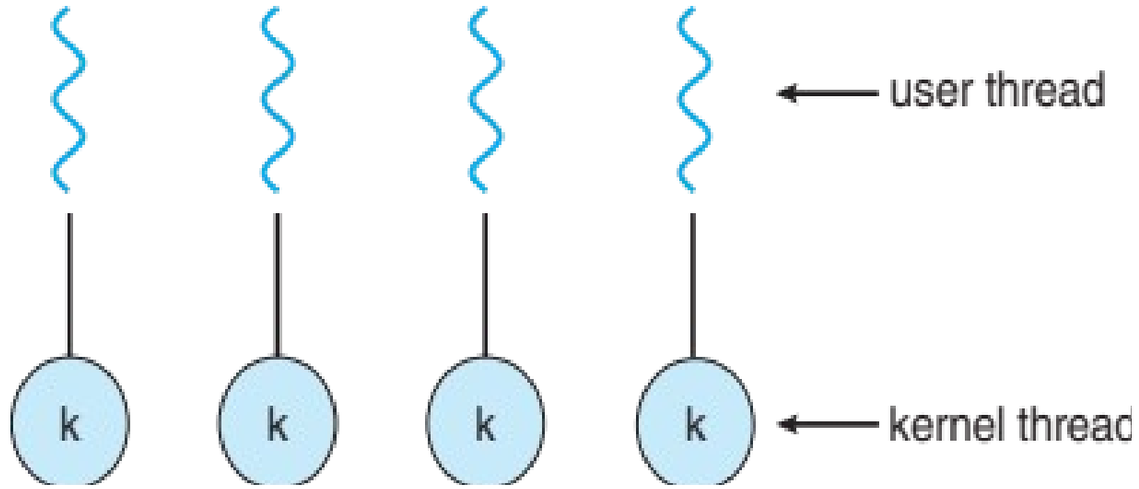

Fig 6. One-to-One Model [1]

Many-to-many Model

- Allows many user level threads to be mapped to many kernel threads
- Allows the operating system to create a sufficient number of kernel threads
- Solaris prior to version 9
- Windows with the ThreadFiber package

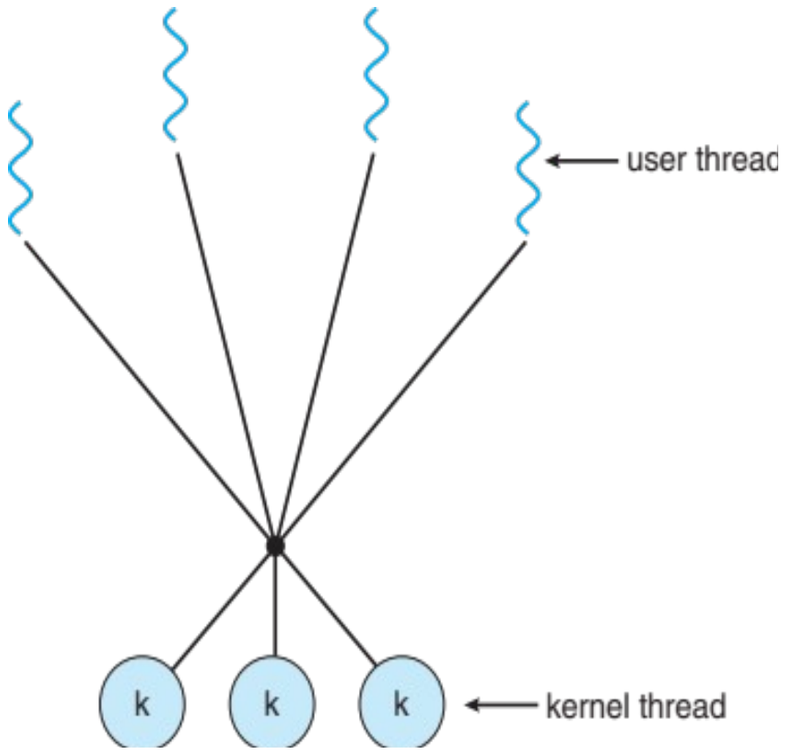

Fig 7. Many-to-Many Model [1]

Two-Level Model

- Similar to M:M, except that it allows a user thread to be bound to kernel thread
- Examples
  - IRIX
  - HP-UX
  - Tru64 UNIX
  - Solaris 8 and earlier

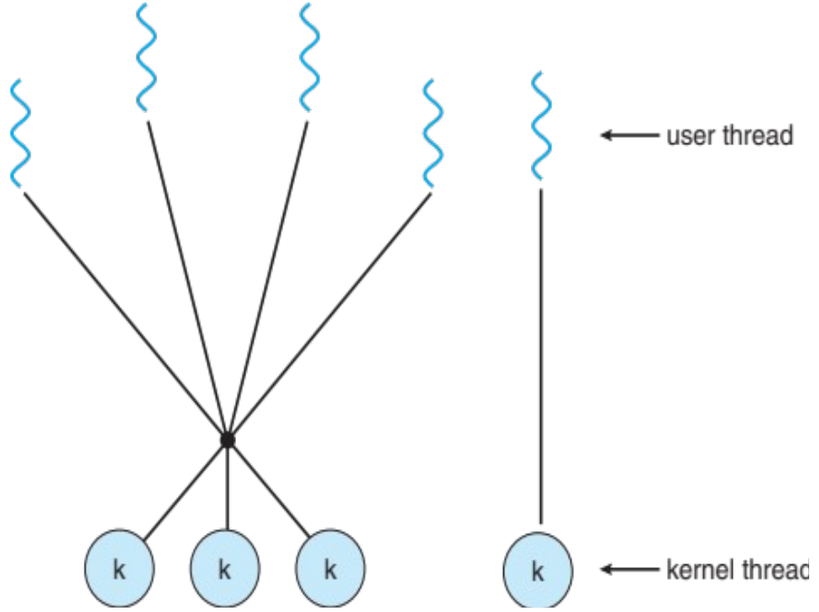

Fig 8. Two-Level Model [1]

Difference between multithreading models

Aspect	Many-to-One Model	One-to-One Model	Many-to-Many Model
Description	Maps many user-level threads to one kernel thread.	Maps each user-level thread to a kernel thread.	Maps many user-level threads to many kernel threads.
Concurrency	Limited concurrency; only one thread can access the kernel at a time.	High concurrency; multiple threads can run in parallel on multiple processors.	Flexible concurrency; can handle more threads than kernel threads.
Advantages	Simple implementation; minimal overhead.	Allows true parallelism on multiprocessors; each thread can be independently scheduled.	Avoids the limitations of both many-to-one and one-to-one models; ent thread management.
Disadvantages	Blocking of one thread blocks all threads; no true parallelism.	Significant overhead for creating and managing many kernel threads.	More complex to implement than other models.
Suitability	Suitable for applications that do not require high concurrency.	Suitable for applications that require high concurrency and true parallelism.	Suitable for applications that need a balance between concurrency and resource utilization.
Examples	Green threads in early versions of Java.	Windows operating system, Linux pthreads.	Solaris operating system.

UPES
UNIVERSITY OF TOMORROW

Summary

- **Threads and Their Benefits:** Explains what threads are and the advantages they offer in concurrent execution.
- **Multi-threading Models:** Discusses different models used for implementing multi-threading.
- **Types of Threads:** Describes various thread types, including kernel-level, user-level, and hybrid threads.
- **Kernel Level Threads:** Provides details about threads managed by the operating system kernel.
- **User Level Threads:** Covers threads managed by user-level libraries.
- **Hybrid Threads:** Combines aspects of both kernel and user-level threading for improved performance and flexibility.

Reference Material

[1]. Silberschatz, A. & Galvin, P. (2009) Operating System Concepts. 8th ed. NJ: John Wiley & Sons, Inc.

- Download Link- https://www.mbit.edu.in/wp-content/uploads/2020/05/Operating_System_Concepts_8th_EditionA4.pdf

[2]. NPTEL Video Lecture: https://onlinecourses.nptel.ac.in/noc24_cs80/preview

UPES
UNIVERSITY OF TOMORROW

MCQ’s

1. Which of the following best defines a thread?
   a) An independent program
   b) A lightweight process within a process
   c) A system call for executing code
   d) A subroutine for handling exceptions

2. What is one of the primary benefits of using threads in a program?
   a) Decreased parallelism
   b) Increased memory usage
   c) Improved responsiveness
   d) Higher code complexity

MCQ’s

3. Which of the following is not a multi-threading model?
   a) Kernel Level thread
   b) User Level thread
   c) Hybrid thread
   d) Sequential thread

4. Which type of thread is managed entirely by the kernel of the operating system?
   a) Kernel Level thread
   b) User Level thread
   c) Hybrid thread
   d) Sequential thread

UPES
UNIVERSITY OF TOMORROW

MCQ's

5. What is a characteristic of User Level threads?
   a) They require less context-switching overhead.
   b) They offer better OS support for thread management.
   c) They are faster in context-switching than Kernel Level threads.
   d) They are not portable across different operating systems.

6. Hybrid threads combine features of which two types of threads?
   a) Kernel Level and User Level threads
   b) Sequential and Concurrent threads
   c) Synchronous and Asynchronous threads
   d) Critical and Non-Critical threads

MCQ's Answers

Answers:

1.b) A lightweight process within a process

2.c) Improved responsiveness

3.d) Sequential thread

4.a) Kernel Level thread

5.a) They require less context-switching overhead.

6.a) Kernel Level and User Level threads

UPES
UNIVERSITY OF TOMORROW

What's Next

1. Thread Scheduling
    1. Content Scope
    2. Pthread Scheduling
    3. Threading Issues

Thank You

UPES
UNIVERSITY OF TOMORROW