SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

23CST202 – Operating Systems

II YEAR - IV SEM

UNIT 1 – Overview and Process Management

TOPIC 3 – Threads

Syllabus

UNIT I OVERVIEW AND PROCESS MANAGEMENT 9

Introduction - Computer System Organization, Architecture, Operation, Process Management – Memory Management – Storage Management – Operating System – Process concept – Process scheduling – Operations on processes – Cooperating processes – Inter process communication. Threads - Multi-threading Models – Threading issues.

UNIT II PROCESS SCHEDULING AND SYNCHRONIZATION 10

CPU Scheduling - Scheduling criteria – Scheduling algorithms – Multiple-processor scheduling – Real time scheduling – Algorithm Evaluation. Process Synchronization - The critical-section problem – Synchronization hardware – Semaphores – Classical problems of synchronization. Deadlock - System model – Deadlock characterization – Methods for handling deadlocks – Deadlock prevention – Deadlock avoidance – Deadlock detection – Recovery from deadlock.

UNIT III MEMORY MANAGEMENT 9

Memory Management - Background – Swapping – Contiguous memory allocation – Paging – Segmentation – Segmentation with paging. Virtual Memory - Background – Demand paging – Process creation – Page replacement – Allocation of frames – Thrashing.

UNIT IV FILE SYSTEMS 8

File concept - Access methods – Directory structure – Files System Mounting – File Sharing – Protection. File System Implementation - Directory implementation – Allocation methods – Free-space management.

UNIT V I/O SYSTEMS 9

I/O Systems - I/O Hardware – Application I/O interface – Kernel I/O subsystem – Streams – Performance. Mass-Storage Structure: Disk scheduling – Disk management – Swap-space management – RAID – Disk attachment – Stable storage – Tertiary storage. Case study: Implementation of Distributed File system in Cloud OS / Mobile OS.

L :45 P:0 T: 45 PERIODS

SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

23CST202 – Operating Systems

II YEAR - IV SEM

CASE STUDY

TOPIC – Threads

Java Threads

- A thread is a single sequence stream within a process.
- Threads are also called lightweight processes as they possess some of the properties of processes.
- Each thread belongs to exactly one process.
- In an operating system that supports multithreading, the process can consist of many threads.
- But threads can be effective only if the CPU is more than 1 otherwise two threads have to context switch for that single CPU.

Java Threads

- All threads belonging to the same process share – code section, data section, and OS resources (e.g. open files and signals)

- But each thread has its own (thread control block) – thread ID, program counter, register set, and a stack

- Any operating system process can execute a thread.

- we can say that single process can have multiple threads.

Why Do We Need Thread?

- Threads run in concurrent manner that improves the application performance.
- Each such thread has its own CPU state and stack, but they share the address space of the process and the environment.
- For example, when we work on Microsoft Word or Google Docs, we notice that while we are typing, multiple things happen together (formatting is applied, page is changed and auto save happens).
- Threads can share common data so they do not need to use inter-process communication.
- Like the processes, threads also have states like ready, executing, blocked, etc.

Why Do We Need Thread?

- Priority can be assigned to the threads just like the process, and the highest priority thread is scheduled first.
- Each thread has its own Thread Control Block (TCB).
- Like the process, a context switch occurs for the thread, and register contents are saved in (TCB).
- As threads share the same address space and resources, synchronization is also required for the various activities of the thread.

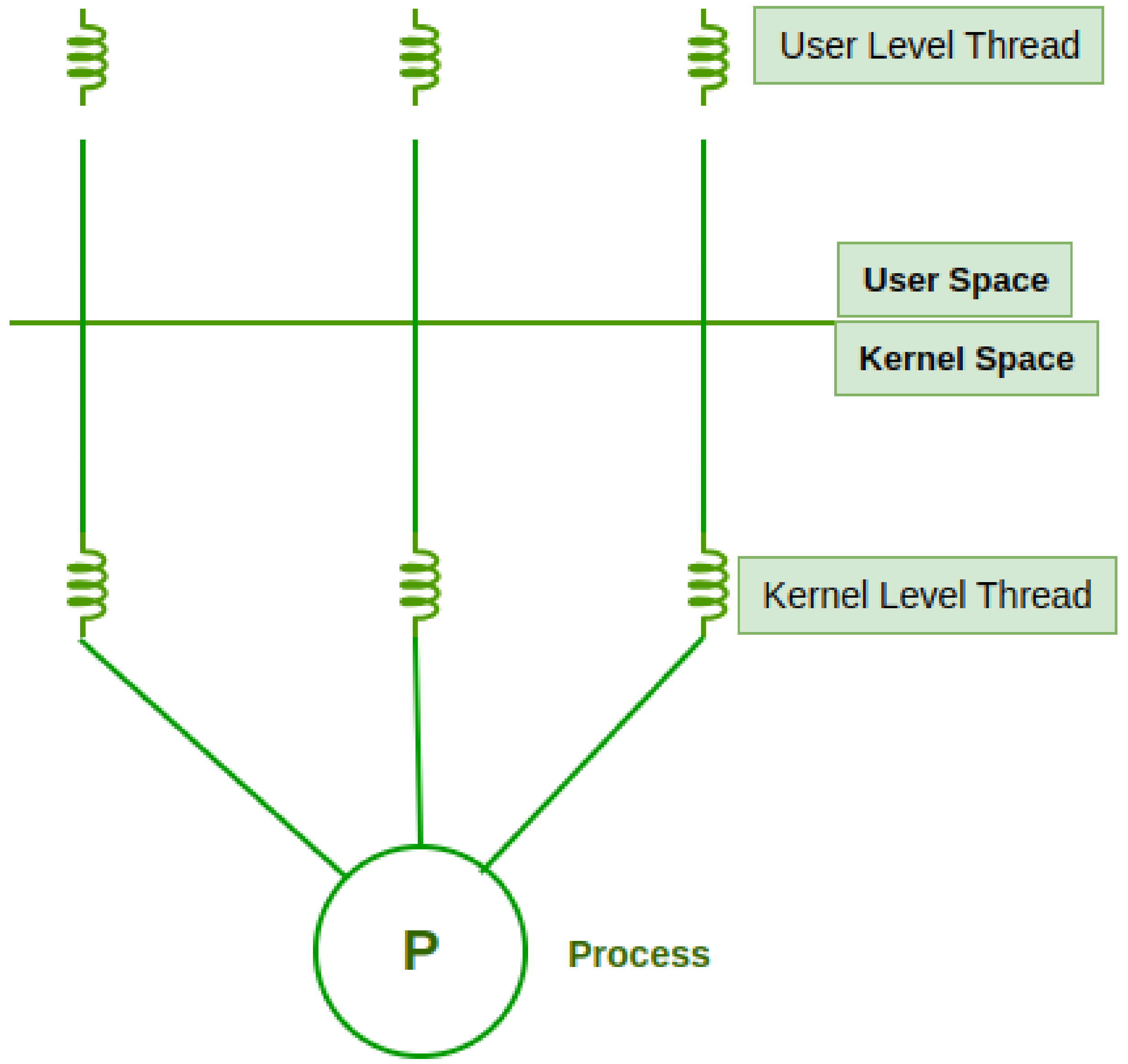
Components of Threads

- These are the basic components of the Operating System.
- **Stack Space:** Stores local variables, function calls, and return addresses specific to the thread.
- **Register Set:** Hold temporary data and intermediate results for the thread’s execution.
- **Program Counter:** Tracks the current instruction being executed by the thread.

Components of Threads

- Types of Thread in Operating System
- Threads are of two types.
- These are described below.

- User Level Thread
- Kernel Level Thread



1. User Level Thread

- User Level Thread is a type of thread that is not created using system calls.
- The kernel has no work in the management of user-level threads.
- User-level threads can be easily implemented by the user.
- In case when user-level threads are single-handed processes, kernel-level thread manages them.
- Let's look at the advantages and disadvantages of User-Level Thread.

2. Kernel Level Threads

- A kernel Level Thread is a type of thread that can recognize the Operating system easily.
- Kernel Level Threads has its own thread table where it keeps track of the system.
- The operating System Kernel helps in managing threads.
- Kernel Threads have somehow longer context switching time.
- Kernel helps in the management of threads.

Benefits of Thread in Operating System

- **Responsiveness**: If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.

- **Faster context switch**: Context switch time between threads is lower compared to the process context switch. Process context switching requires more overhead from the CPU.

- **Effective utilization of multiprocessor system:** If we have multiple threads in a single process, then we can schedule multiple threads on multiple processors. This will make process execution faster.

Benefits of Thread in Operating System

- **Resource sharing:** Resources like code, data, and files can be shared among all threads within a process. Note: Stacks and registers can't be shared among the threads. Each thread has its own stack and registers.
- **Communication**: Communication between multiple threads is easier, as the threads share a common address space. while in the process we have to follow some specific communication techniques for communication between the two processes.
- **Enhanced throughput of the system:** If a process is divided into multiple threads, and each thread function is considered as one job, then the number of jobs completed per unit of time is increased, thus increasing the throughput of the system.

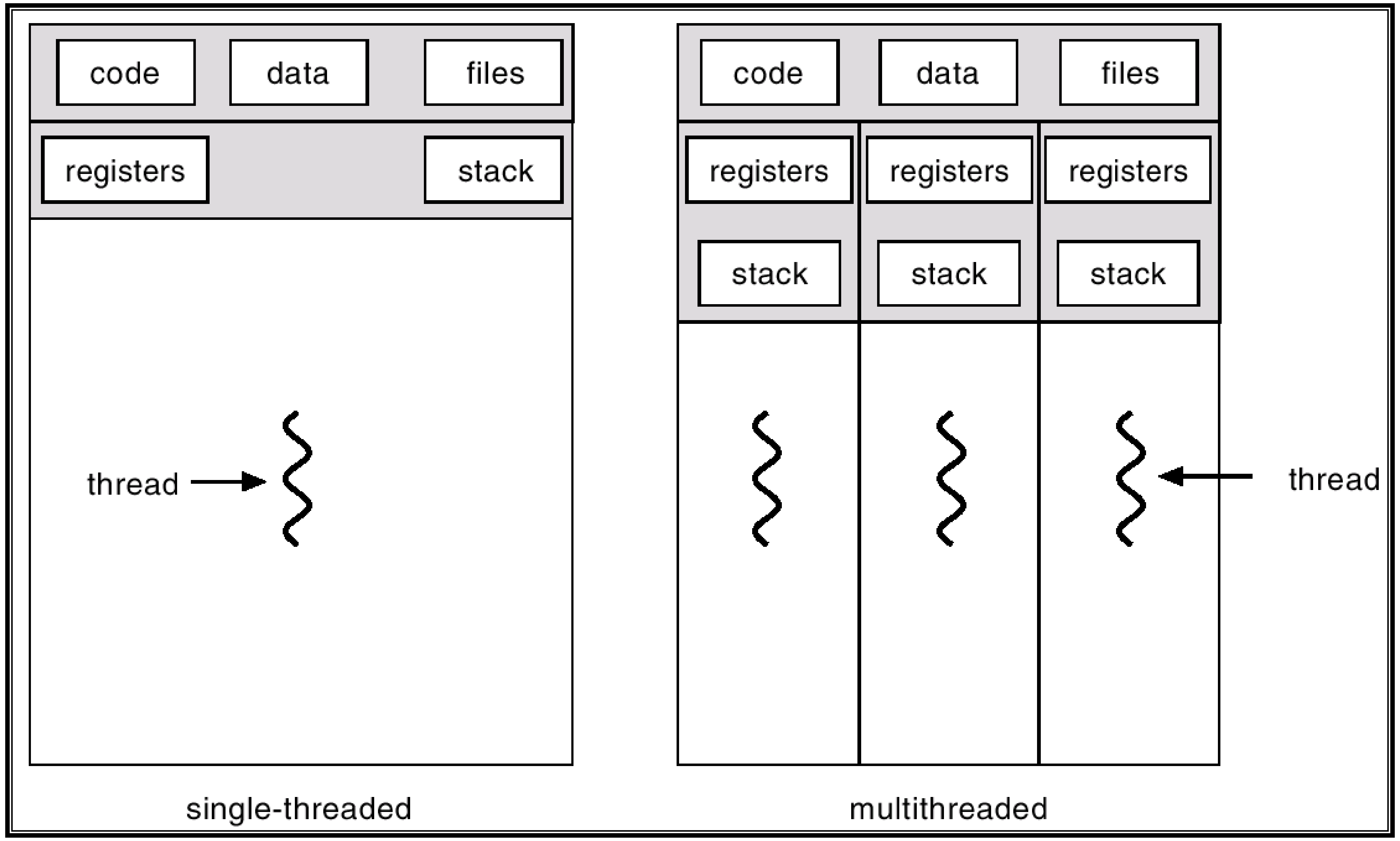
What is Multi-Threading?

- A thread is also known as a lightweight process.
- The idea is to achieve parallelism by dividing a process into multiple threads.
- For example, in a browser, multiple tabs can be different threads.
- MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below.

- Multithreading is a technique used in operating systems to improve the performance and responsiveness of computer systems.
- Multithreading allows multiple threads (i.e., lightweight processes) to share the same resources of a single process, such as the CPU, memory, and I/O devices.

Threads

- Single and Multithreaded Processes



Threads - Benefits

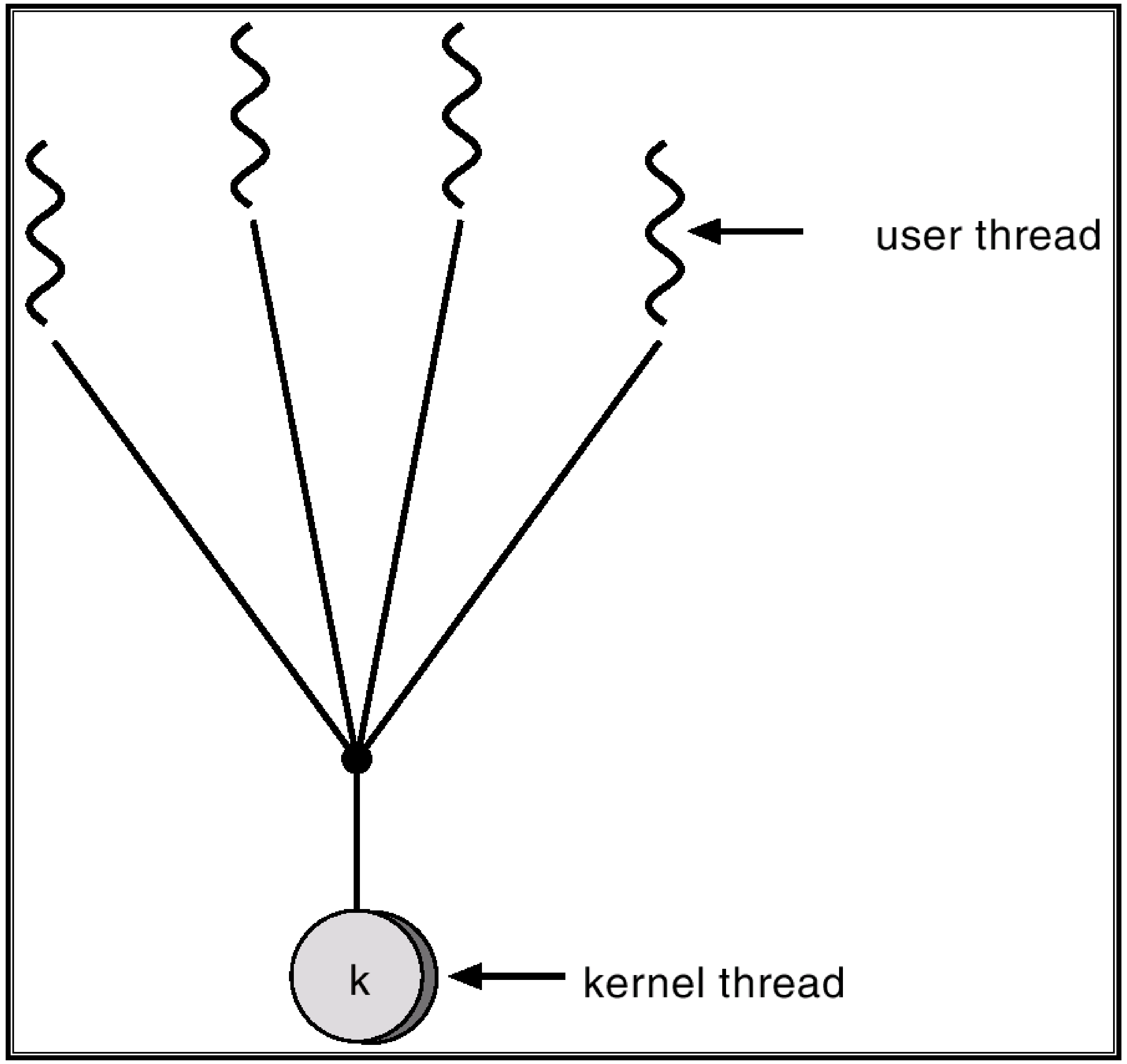
- Responsiveness
- Resource Sharing
- Economy
- Utilization of MP Architectures

Multithreading Models

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

Many-to-One

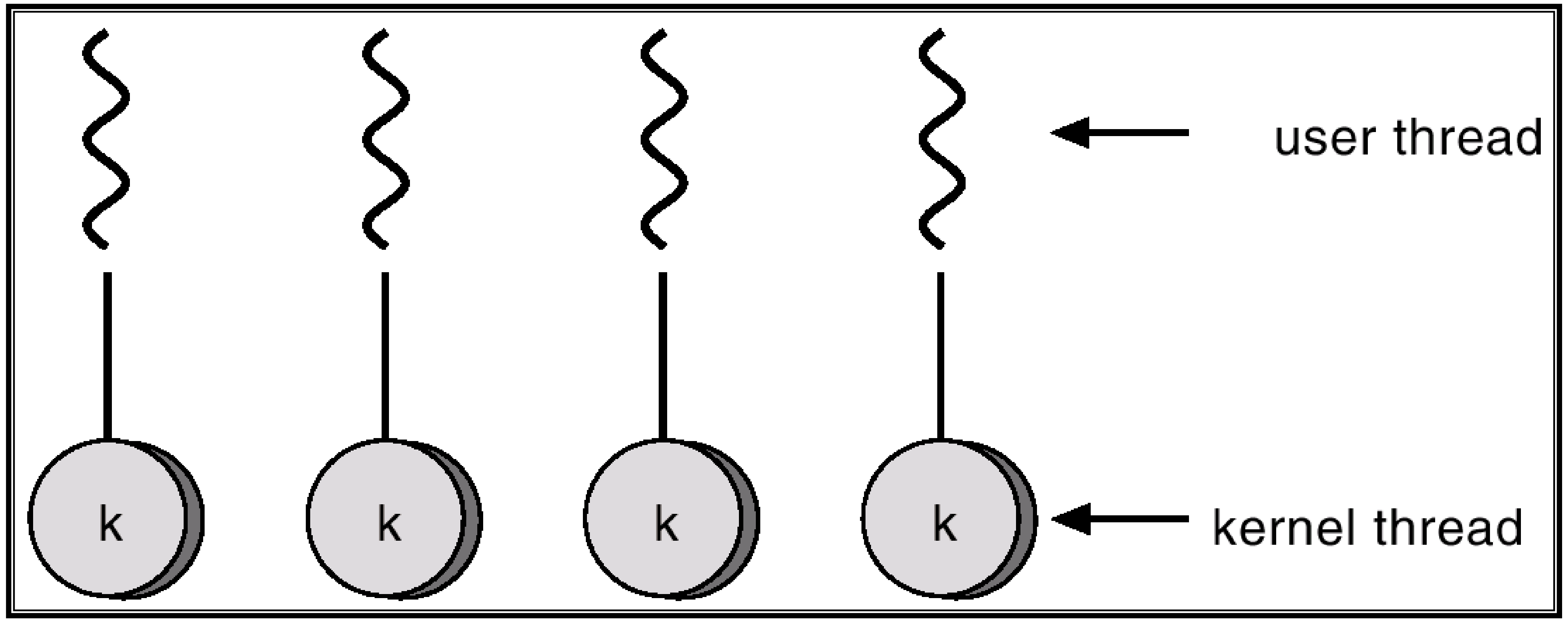
- Many user-level threads mapped to single kernel thread.
- Used on systems that do not support kernel threads.



One-to-One

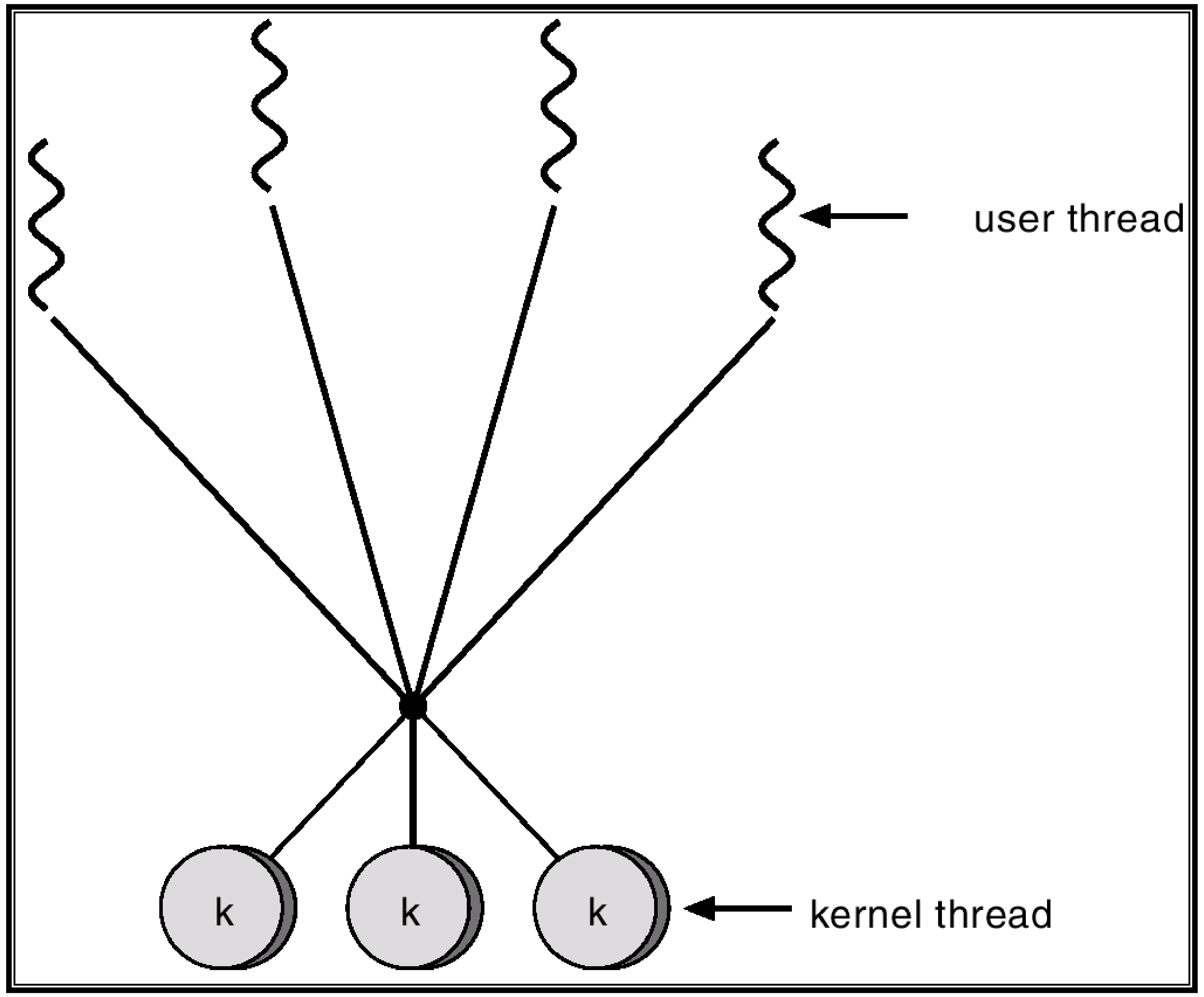
- Each user-level thread maps to kernel thread.

- Examples
-   - Windows 95/98/NT/2000
-   - OS/2



Many-to-Many Model

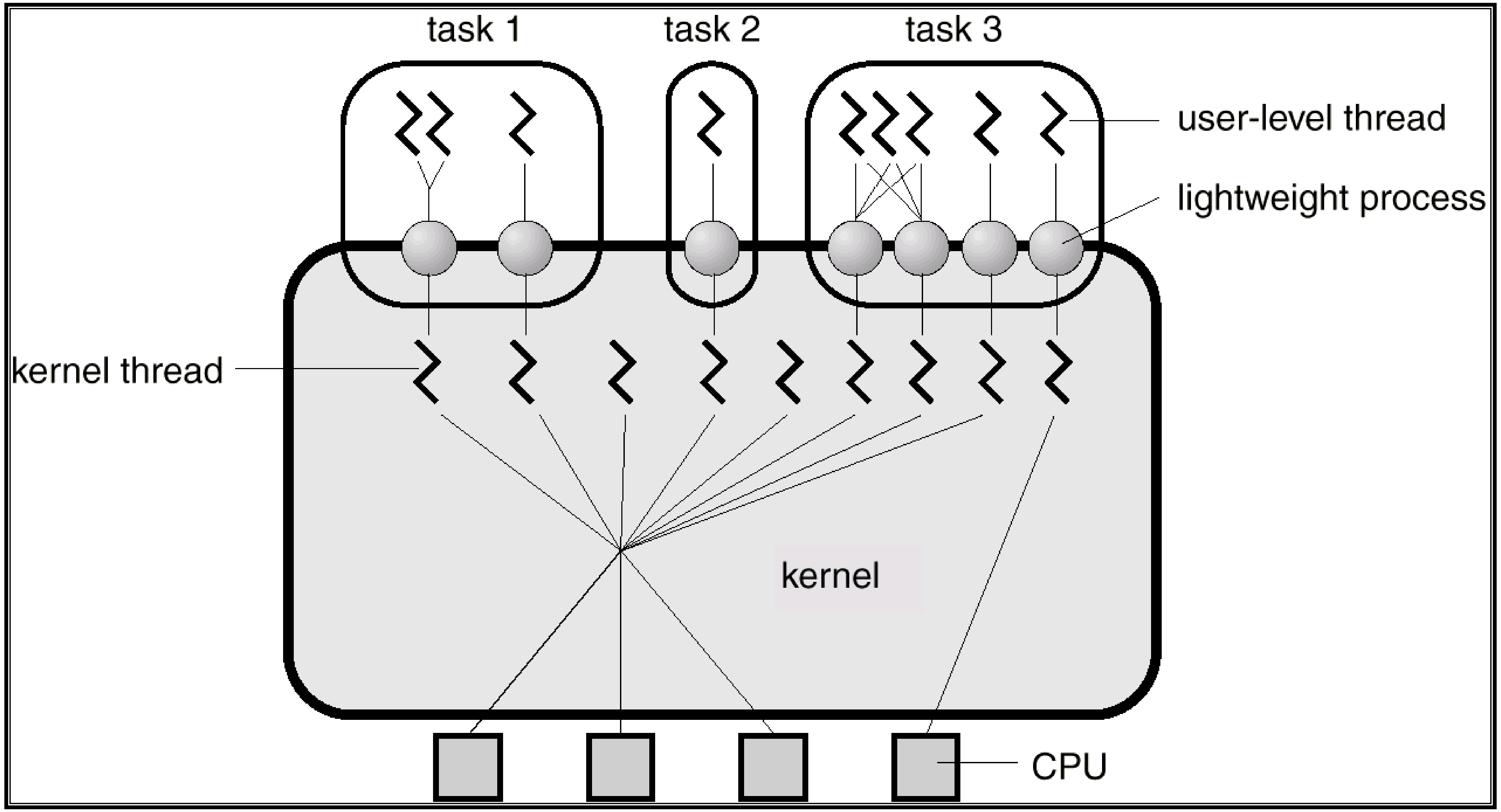
- Allows many user level threads to be mapped to many kernel threads.
- Allows the operating system to create a sufficient number of kernel threads.
- Solaris 2, Windows NT/2000 with the ThreadFiber package



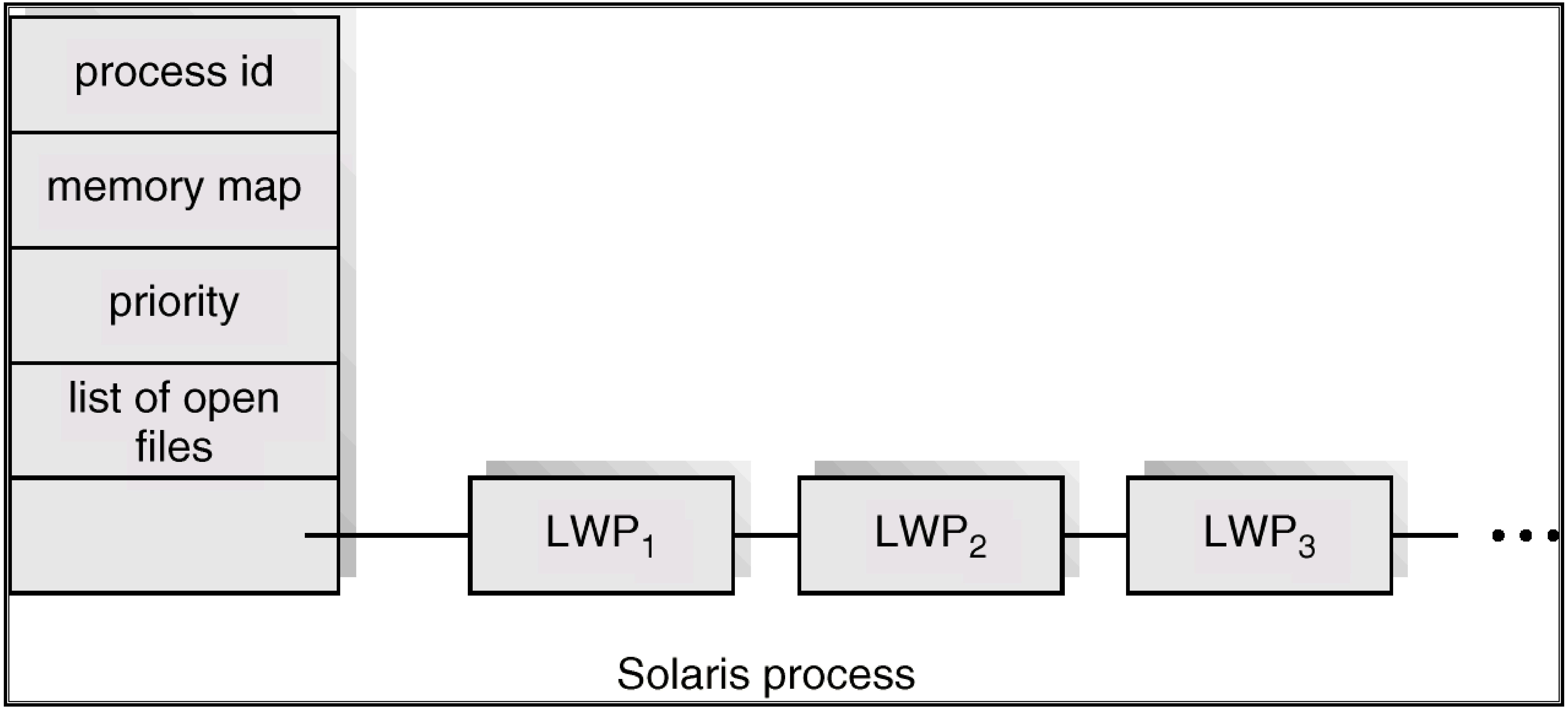
Threading Issues

- Semantics of fork() and exec() system calls.
- Thread cancellation.
- Signal handling
- Thread pools
- Thread specific data

Solaris 2 Threads



Solaris Process



Windows 2000 Threads

- Implements the one-to-one mapping.
- Each thread contains
- - a thread id
- - register set
- - separate user and kernel stacks
- - private data storage area

Linux Threads

- Linux refers to them as tasks rather than threads.
- Thread creation is done through clone() system call.
- Clone() allows a child task to share the address space of the parent task (process)

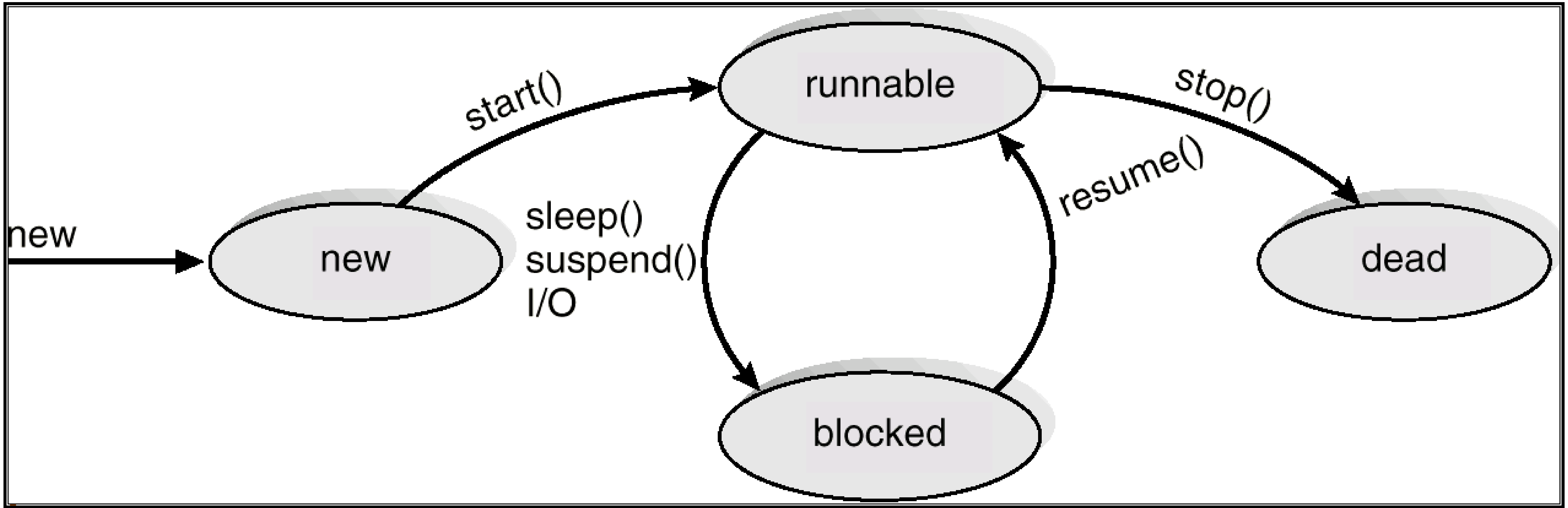
Java Threads

- Java threads may be created by:

  - Extending Thread class
  - Implementing the Runnable interface

- Java threads are managed by the JVM.

Java Thread states


new
new
start()
runnable
stop()
dead
sleep()
suspend()
I/O
resume()
blocked