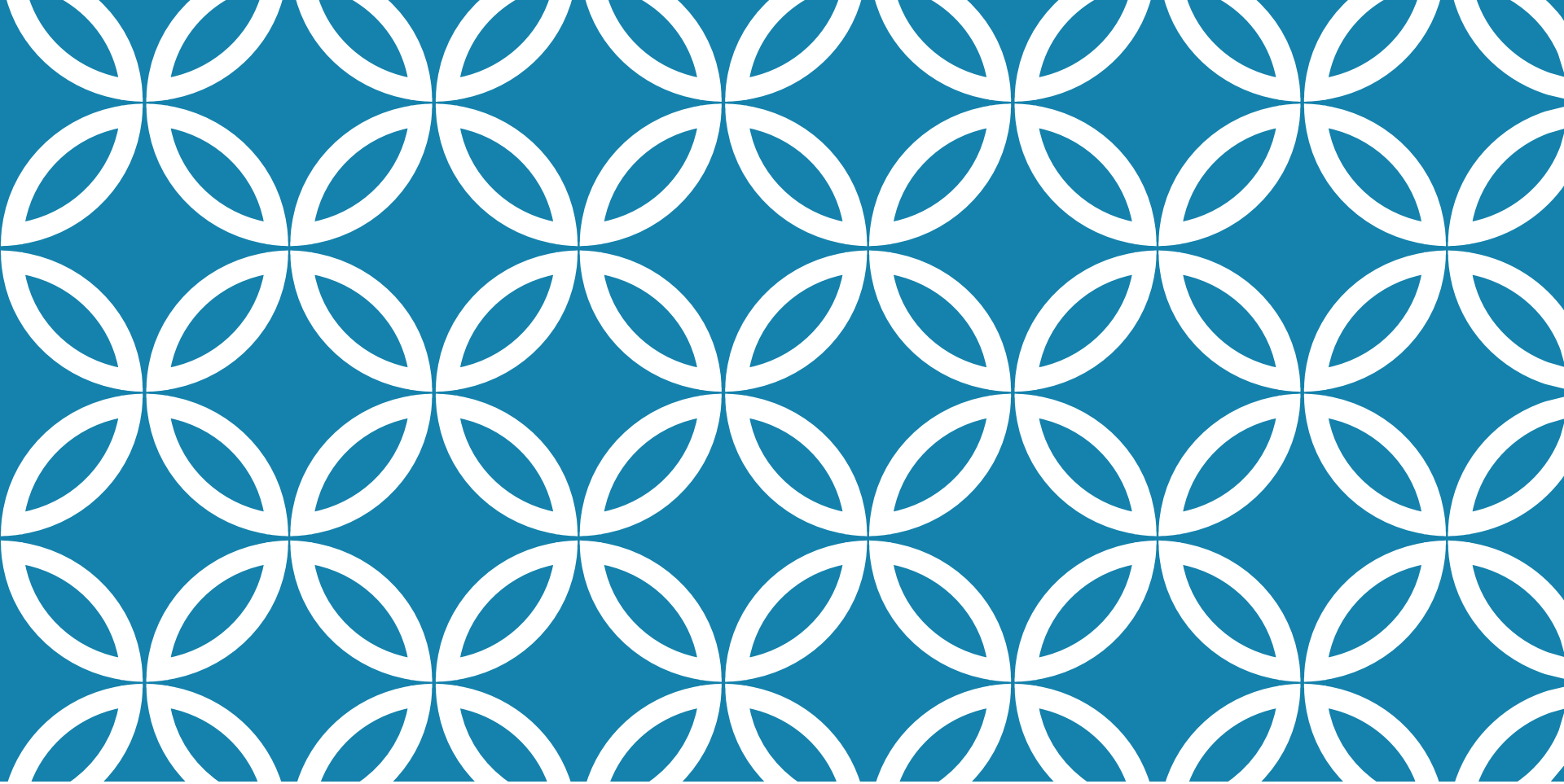


Module II, Part II

PROCESS MANAGEMENT,

OPERATING SYSTEM

Part-II

Threads

Prof. Thaksen Parvat

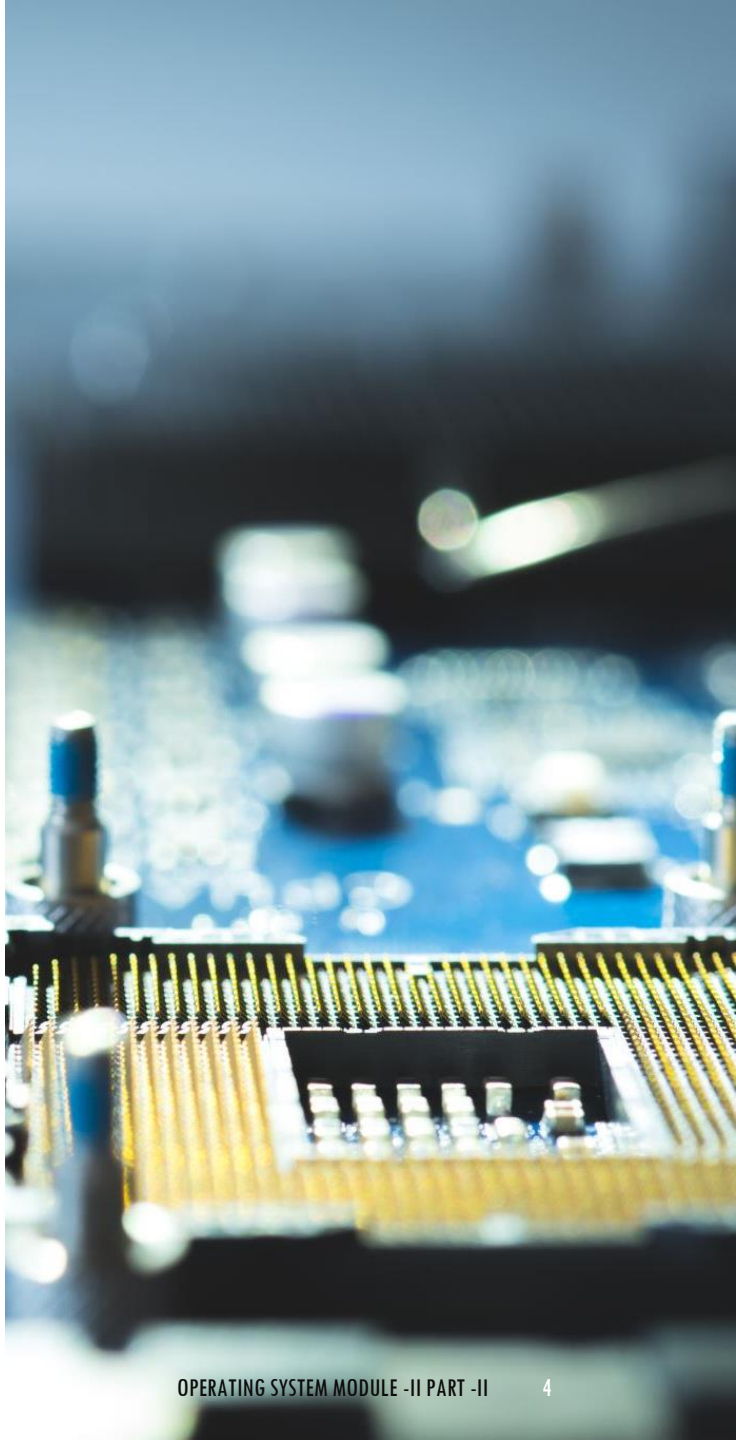
CHAPTER OUTCOMES

After Studying this module, you will be able to;

- Define Threads.
- Understand the distinction between process and thread.
- Explain Types of Threads.
- Understand Thread Models.

THREADS

- A thread is a basic unit of CPU utilization.
- It comprises a thread ID, a PC, a register set, and a stack.
- It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals.



THREADS

- A traditional (or **heavyweight**)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.

THREADS

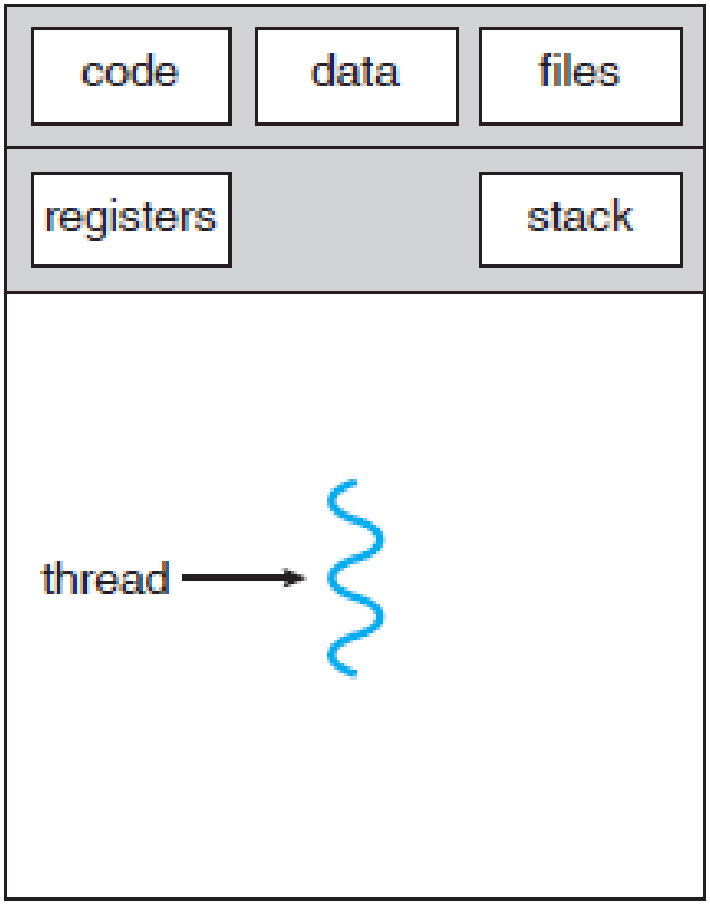


Fig. Single-threaded process
ref: SILBERSCHATZ

THREADS

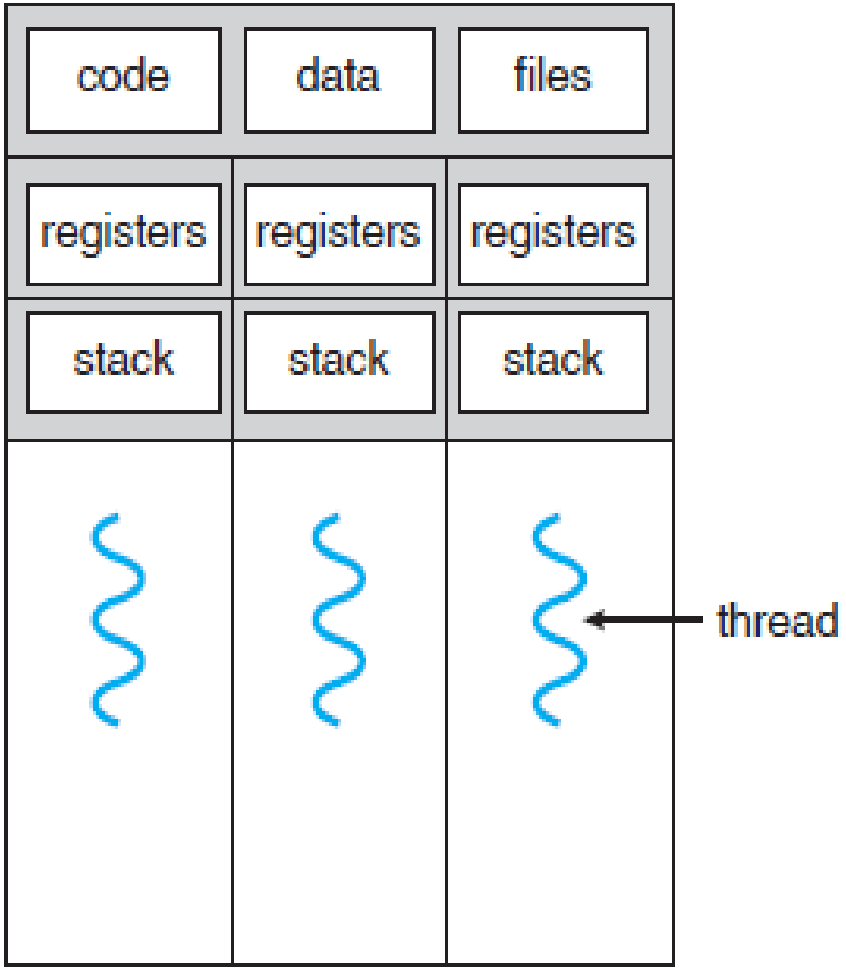


Fig. Multithreaded **process**
ref: SILBERSCHATZ

THREADS

- Many software packages that run on modern desktop PCs are multithreaded.

THREADS

Many software packages that run on modern desktop PCs are multithreaded.

An application typically is implemented as a separate process with several threads of control.

THREADS

Many software packages that run on modern desktop PCs are multithreaded.

An application typically is implemented as a separate process with several threads of control.

A Web browser might have one thread display images or text while another thread retrieves data from the network.

THREADS

Many software packages that run on modern desktop PCs are multithreaded.

An application typically is implemented as a separate process with several threads of control.

A Web browser might have one thread display images or text while another thread retrieves data from the network.

A word processor may have a thread for displaying graphics, another thread for responding to keystrokes from the user, and a third thread for performing spelling and grammar checking in the background.

THREADS

A Web server accepts client requests for Web pages, images, sound, and so forth.

THREADS

A Web server accepts client requests for Web pages, images, sound, and so forth.

A busy Web server may have several (perhaps thousands of) clients concurrently accessing it.

THREADS

A Web server accepts client requests for Web pages, images, sound, and so forth.

A busy Web server may have several (perhaps thousands of) clients concurrently accessing it.

If the Web server ran as a traditional single-threaded process, it would be able to service only one client at a time.

THREADS

If the Web-server process is multithreaded, the server will create a separate thread that listens for client requests.

The server will create a new thread to service the request and resume listening for additional requests.

THREADS

If the Web-server process is multithreaded, the server will create a separate thread that listens for client requests.

The server will create a new thread to service the request and resume listening for additional requests.

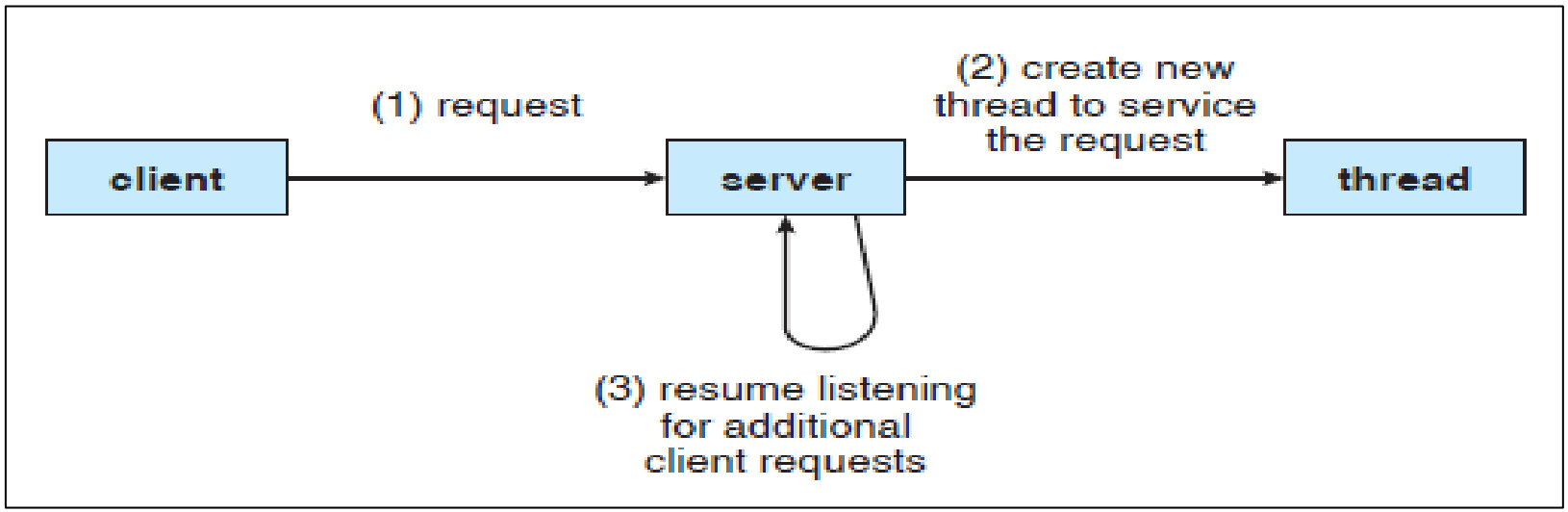


Fig.: Multithreaded server architecture ref: SILBERSCHATZ

BENEFITS OF THREADS

The benefits of multithreaded programming can be broken down into four major categories:

BENEFITS OF THREADS

The benefits of multithreaded programming can be broken down into four major categories:

1. Responsiveness
2. Resource sharing
3. Economy
4. Scalability

BENEFITS OF THREADS

1. Responsiveness:

- Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.

BENEFITS OF THREADS

1. Responsiveness:

- Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.

- For instance, a multithreaded Web browser could allow user interaction in one thread while an image was being loaded in another thread.

BENEFITS OF THREADS

2. Resource sharing:

- Processes may only share resources through techniques such as shared memory or message passing.
- Such techniques must be explicitly arranged by the programmer.
- However, threads share the memory and the resources of the process to which they belong by default.
- The benefit of sharing code and data is that it allows an application to have several different threads of activity within the same address space.

BENEFITS OF THREADS

3. Economy:

- Allocating memory and resources for process creation is costly.
- Because threads share the resources of the process to which they belong, it is more economical to create and context-switch threads.
- Empirically gauging the difference in overhead can be difficult, but in general it is much more time consuming to create and manage processes than threads.

BENEFITS OF THREADS

4. Scalability:

- The benefits of multithreading can be greatly increased in a multiprocessor architecture, where threads may be running in parallel on different processors.
- A single-threaded process can only run on one processor, regardless how many are available.
- Multithreading on a multi-CPU machine increases parallelism.

TYPES OF THREADS

There are two broad categories of thread implementation;

- User Level Threads (ULT)
- Kernel Level Threads (KLT)

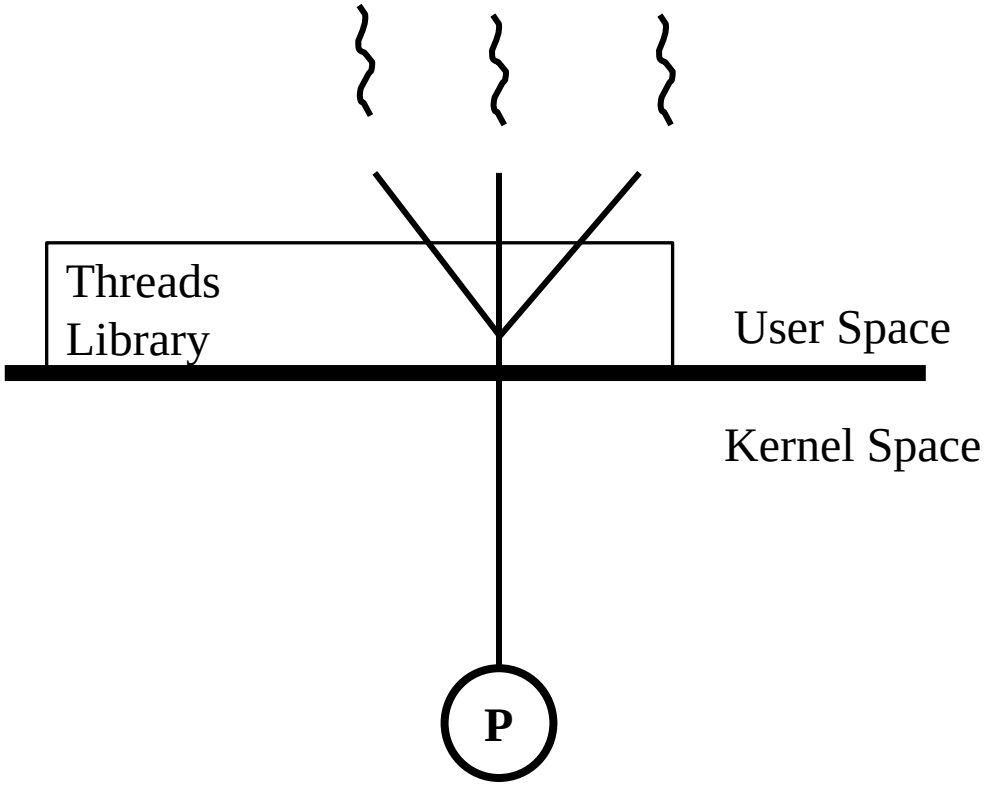
TYPES OF THREADS

User Level Threads (ULT):

- In pure ULT facility, all the work of thread management is done by the application and the kernel is not aware of the existence of thread.

TYPES OF THREADS

User Level Threads (ULT):



TYPES OF THREADS

User Level Threads (ULT):

- The threads library is contains code for;

TYPES OF THREADS

User Level Threads (ULT):

- The threads library is contains code for
  - ✓ Creating and destroying threads,

TYPES OF THREADS

User Level Threads (ULT):

- The threads library contains code for
  - ✓ Creating and destroying threads,
  - ✓ Passing messages and data between threads,

TYPES OF THREADS

User Level Threads (ULT):

- The threads library is contains code for
  - ✓ Creating and destroying threads,
  - ✓ Passing messages and data between threads,
  - ✓ Scheduling thread execution,

TYPES OF THREADS

User Level Threads (ULT):

- The threads library is contains code for
  - ✓ Creating and destroying threads,
  - ✓ Passing messages and data between threads,
  - ✓ Scheduling thread execution,
  - ✓ Saving and restoring thread context.

TYPES OF THREADS

User Level Threads (ULT):

- By default, an application begins with a single thread and begins running with that thread.
- The application may spawn a new thread to run within the same process.

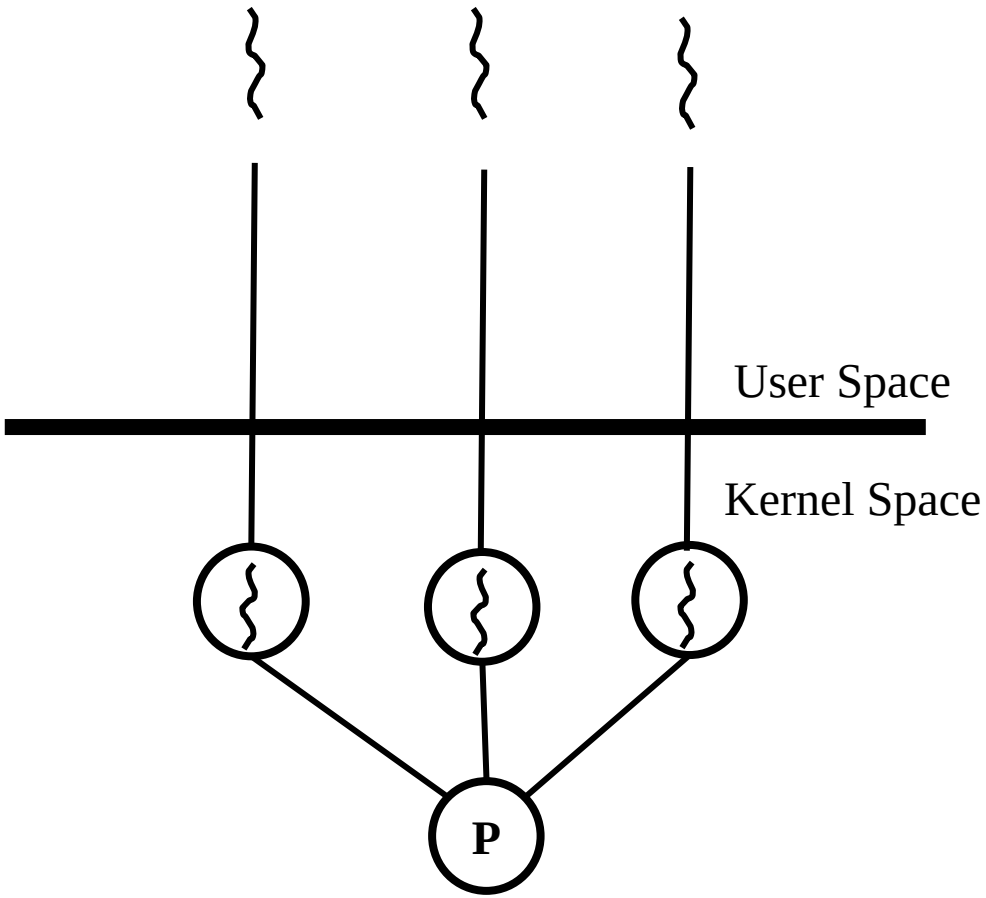
TYPES OF THREADS

Kernel Level Threads (KLT)

- In a pure KLT facility, all the work of thread management done by the kernel.
- There is no thread management code in application level, simply an application programming interface (API) to the kernel thread facility.

TYPES OF THREADS

Kernel Level Threads (KLT)



TYPES OF THREADS

Kernel Level Threads (KLT)

- The kernel maintains the context information for the process as a whole and for individual threads within the process.

TYPES OF THREADS

Kernel Level Threads (KLT)

- The kernel maintains the context information for the process as a whole and for individual threads within the process.
- Scheduling by the kernel is done on a thread basis.

TYPES OF THREADS

Kernel Level Threads (KLT)

- The kernel maintains the context information for the process as a whole and for individual threads within the process.
- Scheduling by the kernel is done on a thread basis.
- The kernel can simultaneously schedule multiple threads from the same process on multiple processor.

TYPES OF THREADS

Kernel Level Threads (KLT)

- The kernel maintains the context information for the process as a whole and for individual threads within the process.
- Scheduling by the kernel is done on a thread basis.
- The kernel can simultaneously schedule multiple threads from the same process on multiple processor.
- If one thread in a process is blocked, the kernel can schedule another thread of same process.

TYPES OF THREADS

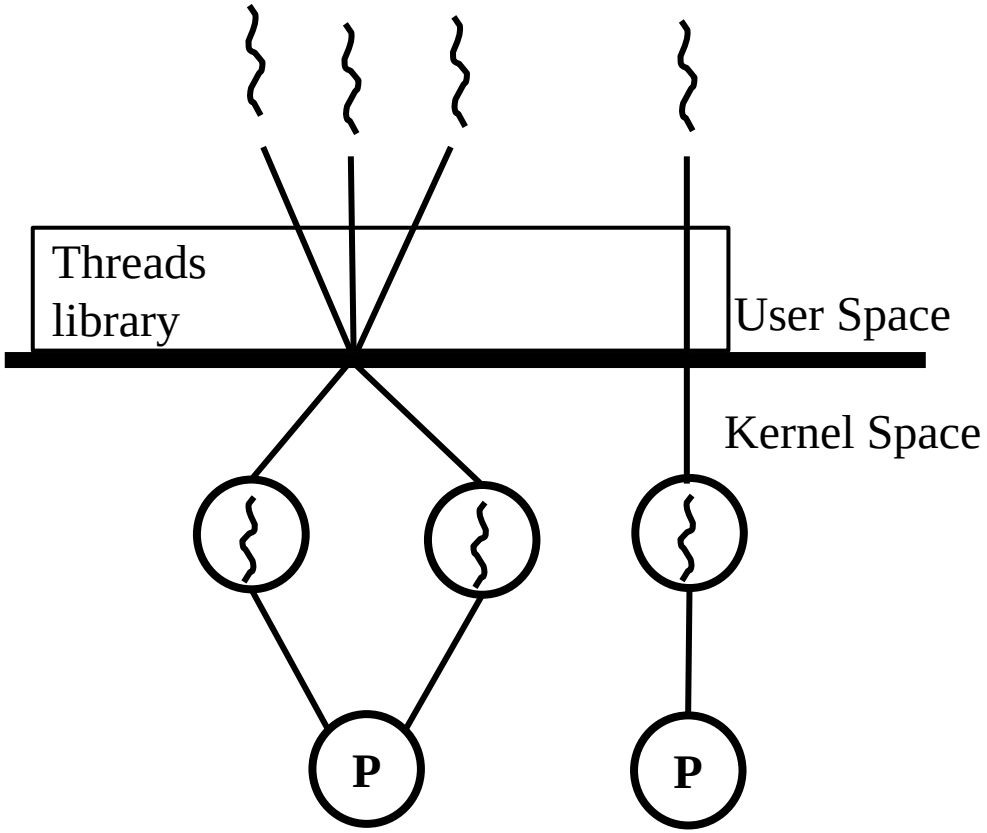
Kernel Level Threads (KLT)

- The disadvantage of KLT over ULT is that the transfer of control from one thread to another is within the same process requires a mode switch to the kernel.

TYPES OF THREADS

Combined Approach (ULT and KLT):

- Some OS provide a combined ULT /KLT facility.



TYPES OF THREADS

Combined Approach (ULT and KLT):

- Threads creation is done completely in user space.
- Multiple ULTs from a single application mapped onto some number of KLTs.

THREAD MODELS

- Support for threads may be provided either at the user level, for **user threads**, or by the kernel, for **kernel threads**.

- User threads are supported above the kernel and are managed without kernel support, whereas kernel threads are supported and managed directly by the operating system.

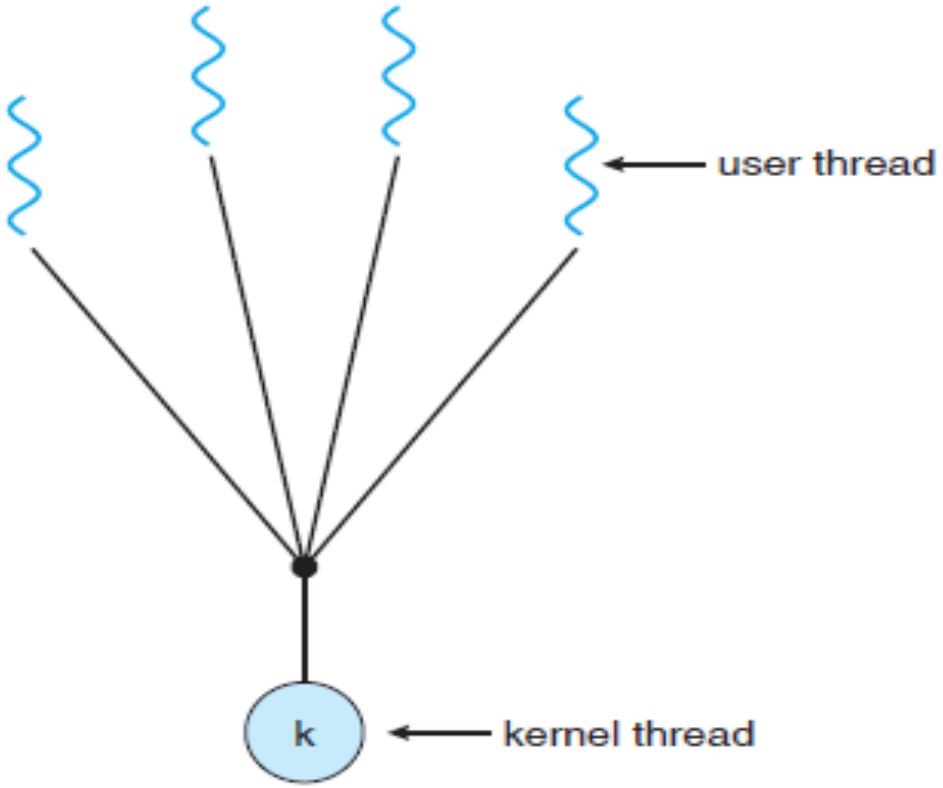
- Virtually all OS - including Windows, Linux, Mac OS X, Solaris, and Tru64 UNIX (formerly Digital UNIX) support kernel threads.

THREAD MODELS

- A relationship must exist between user threads and kernel threads.
- Three common ways of establishing such a relationship.
    1. Many-to-One Model
    2. One-to-One Model
    3. Many-to-Many Model

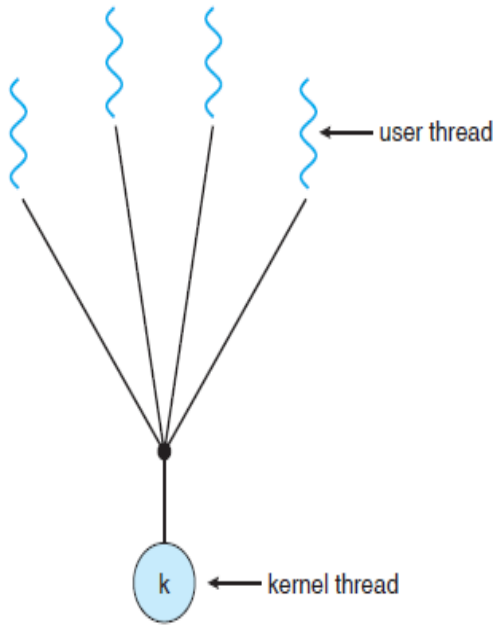
THREAD MODELS

1. Many-to-One Model: The many-to-one model maps many user-level threads to one kernel thread.



THREAD MODELS

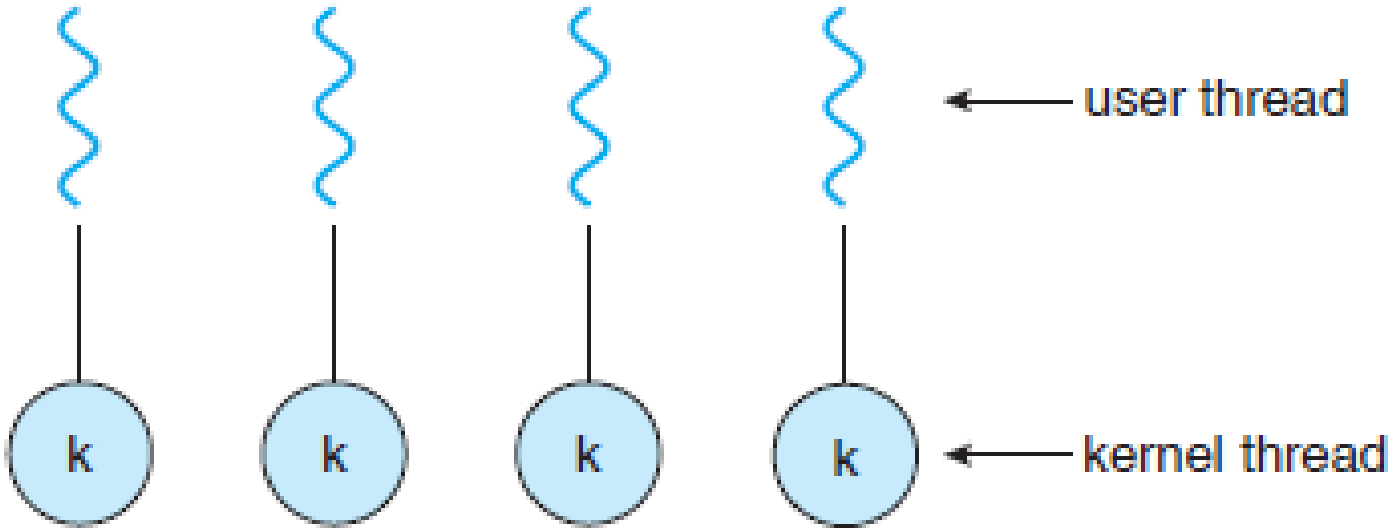
1. Many-to-One Model:



- Thread management is done by the thread library in user space, so it is efficient;
- But the entire process will block if a thread makes a blocking system call.
- Also, because only one thread can access the kernel at a time, multiple threads are unable to run in parallel on multiprocessors.

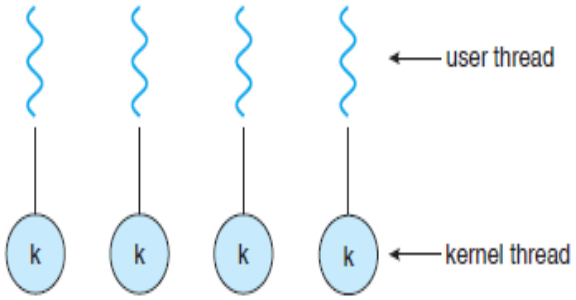
THREAD MODELS

2. One-to-One Model: The one-to-one model maps each user thread to a kernel thread.



THREAD MODELS

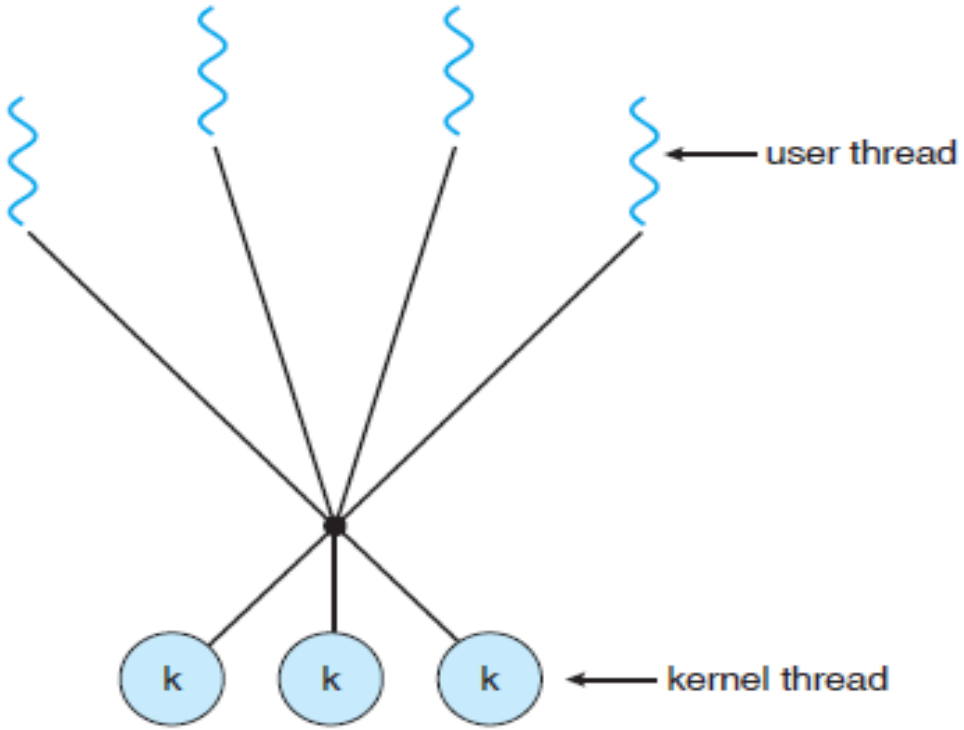
2. One-to-One Model:



- It provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call;
- It also allows multiple threads to run in parallel on multiprocessors.
- The only drawback to this model is that creating a user thread requires creating the corresponding kernel thread.

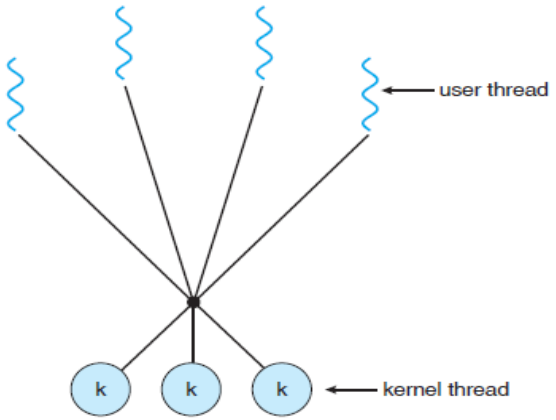
THREAD MODELS

3. Many-to-Many Model: The many-to-many model multiplexes many user-level threads to a smaller or equal number of kernel threads.



THREAD MODELS

3. Many-to-Many Model:



- The number of kernel threads may be specific to either a particular application or a particular machine.

- Developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor.
- Also, when a thread performs a blocking system call, the kernel can schedule another thread for execution.

THREAD MODELS

- A **thread library** provides the programmer with an API for creating and managing threads.
- There are two primary ways of implementing a thread library.

THREAD LIBRARIES

1. The first approach is to provide a library entirely in user space with no kernel support.

- All code and data structures for the library exist in user space.

- This means that invoking a function in the library results in a local function call in user space and not a system call.

THREAD LIBRARIES

2. The second approach is to implement a kernel-level library supported directly by the operating system.

- In this case, code and data structures for the library exist in kernel space.

- Invoking a function in the API for the library typically results in a system call to the kernel.

THREAD LIBRARIES

- Three main thread libraries are in use today:
  - (1) POSIX Pthreads
  - (2) Win32
  - (3) Java

THREAD LIBRARIES

- Three main thread libraries are in use today:
  (1) POSIX Pthreads
  Pthreads, the threads extension of the POSIX standard, may be provided as either a user- or kernel-level library.
  (2) Win32,
  (3) Java.

THREAD LIBRARIES

- Three main thread libraries are in use today:
  - (1) POSIX Pthreads
  - (2) Win32
    The Win32 thread library is a kernel-level library available on Windows systems.
  - (3) Java

THREAD LIBRARIES

- Three main thread libraries are in use today:
  - (1) POSIX Pthreads
  - (2) Win32
  - (3) Java

    The Java thread API allows threads to be created and managed directly in Java programs.

Thank you