THREADS

Gursharan Singh Tatla

professorgstatla@gmail.com

THREAD

- A thread is a single sequential flow of execution of the tasks of a process.
- A thread is a lightweight process and the smallest unit of CPU utilization. Thus, a thread is like a miniprocess.
- Each thread has a thread id, program counter, register set and a stack.
- A thread undergoes different states such as new, ready, running, waiting and terminated similar to that of a process.
- However, a thread is not a program as it cannot run on its own. It runs within a program.

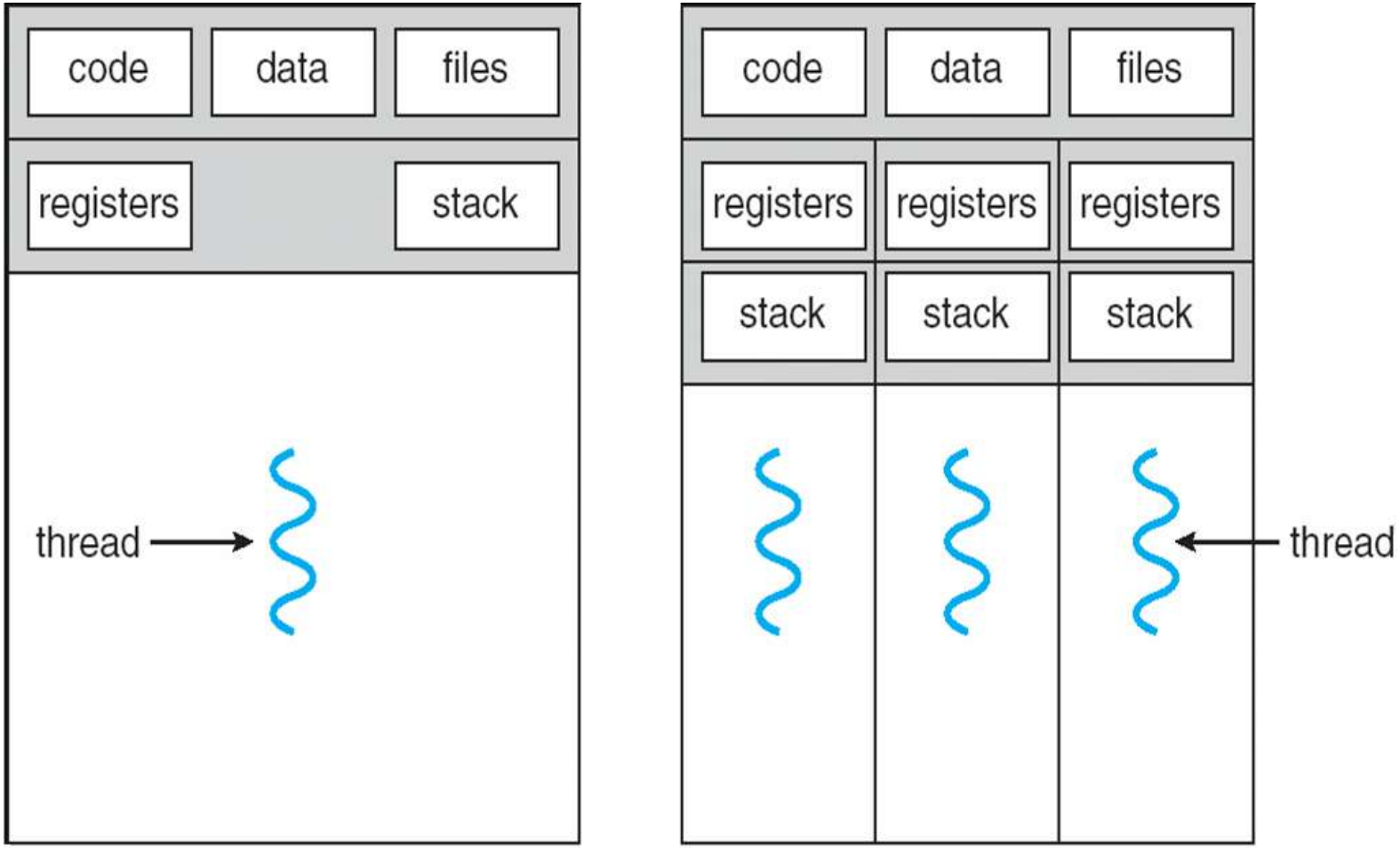
Multi-Threading

- A process can have single thread of control or multiple threads of control.
- If a process has single thread of control, it can perform only one task at a time.
- Many modern operating systems have extended the process concept to allow a process to have multiple threads.
- Thus, allowing the process to perform multiple tasks at the same time.
- This concept is known as **Multi-Threading**.

Multi-Threading

- For e.g.:
  - The tasks in a web browser are divided into multiple threads.
  - Downloading the images, downloading the text and displaying images and text.
  - While one thread is busy in downloading the images, another thread displays it.
- The various operating systems the implement multithreading are Windows XP, Vista, 7, Server 2000 onwards, Linux etc.
- In multithreading, a thread can share its code, data and resources with other threads of same process.

SINGLE THREAD & MULTI-THREAD

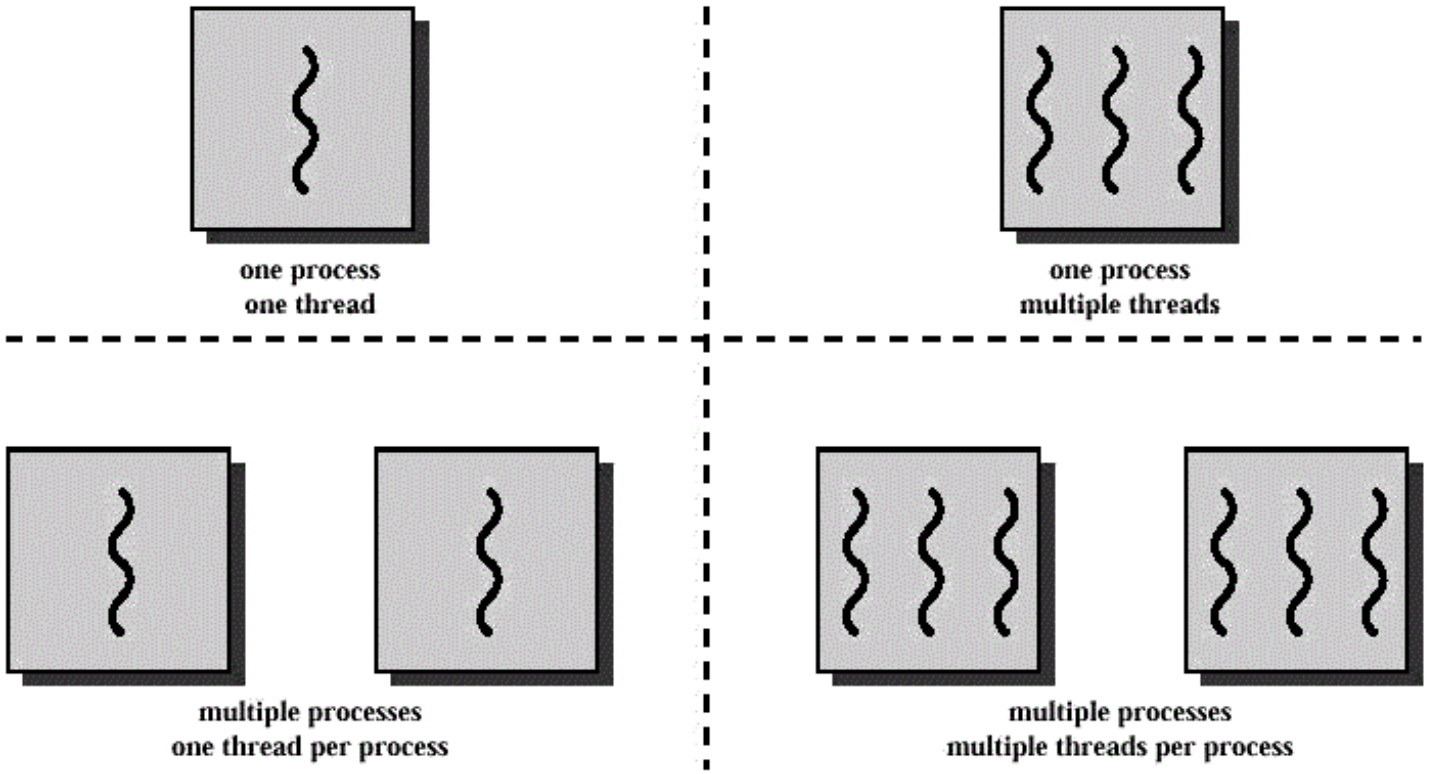


single-threaded process

multithreaded process

Threads & Processes

- An idea of how threads & processes can be related to each other is depicted in the fig.:



Threads & Processes

- There are several similarities and differences between a thread and a process:
  - **Similarities:**
    - Like process, each thread has its own program counter and stack.
    - Threads share CPU just as a process.
    - Threads also run sequentially, like a process.
    - Threads can create child threads.
    - Threads have the same states as process: new, ready, running, waiting and terminated.

Threads & Processes

- **Differences:**
  - Each process has its own distinct address space in the main memory. On the other hand, all threads of a same process share same address space.
  - Threads require less system resources than a process.
  - Threads are not independent of each other, unlike processes.
  - Threads take less time for creation and termination than a process.
  - It takes less time to switch between two threads than to switch between two processes.

Types of Threads

- Threads are of three types:
  - Kernel Level Threads
  - User Level Threads
  - Hybrid Threads

Kernel Level Threads

- Threads of processes defined by operating system itself are called **Kernel Level Threads**.
- In these types of threads, kernel performs thread creation, scheduling and management.
- Kernel threads are used for internal workings of operating system.
- Kernel threads are slower to create and manage.
- The various operating systems that support kernel level threads are: Windows 2000, XP, Solaris 2.

User Level Threads

- The threads of user application process are called **User Level Threads**.
- They are implemented in the user space of main memory.
- User level library (functions to manipulate user threads) is used for thread creation, scheduling and management without any support from the kernel.
- User level threads are fast to create and manage.

Hybrid Threads

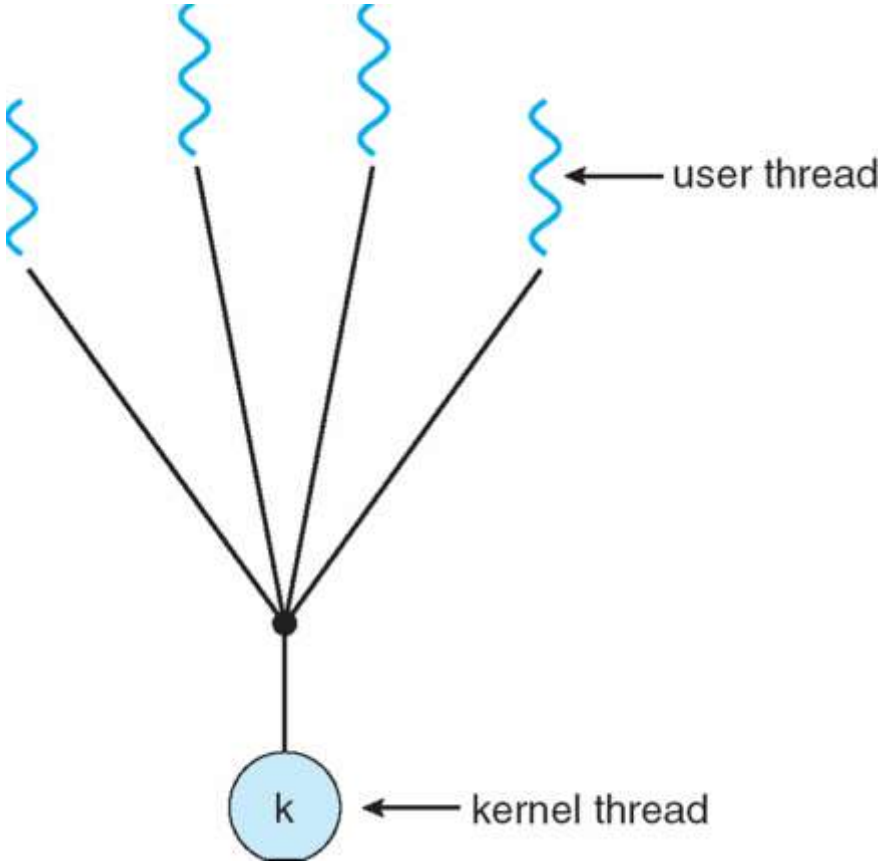
- In hybrid approach, both kernel level threads and user level threads are implemented.
- For e.g.: Solaris 2.

Multi-Threading Models

- Depending on the support for user and kernel threads, there are three multithreading models:
  - Many-to-One Model
  - One-to-One Model
  - Many-to-Many Model

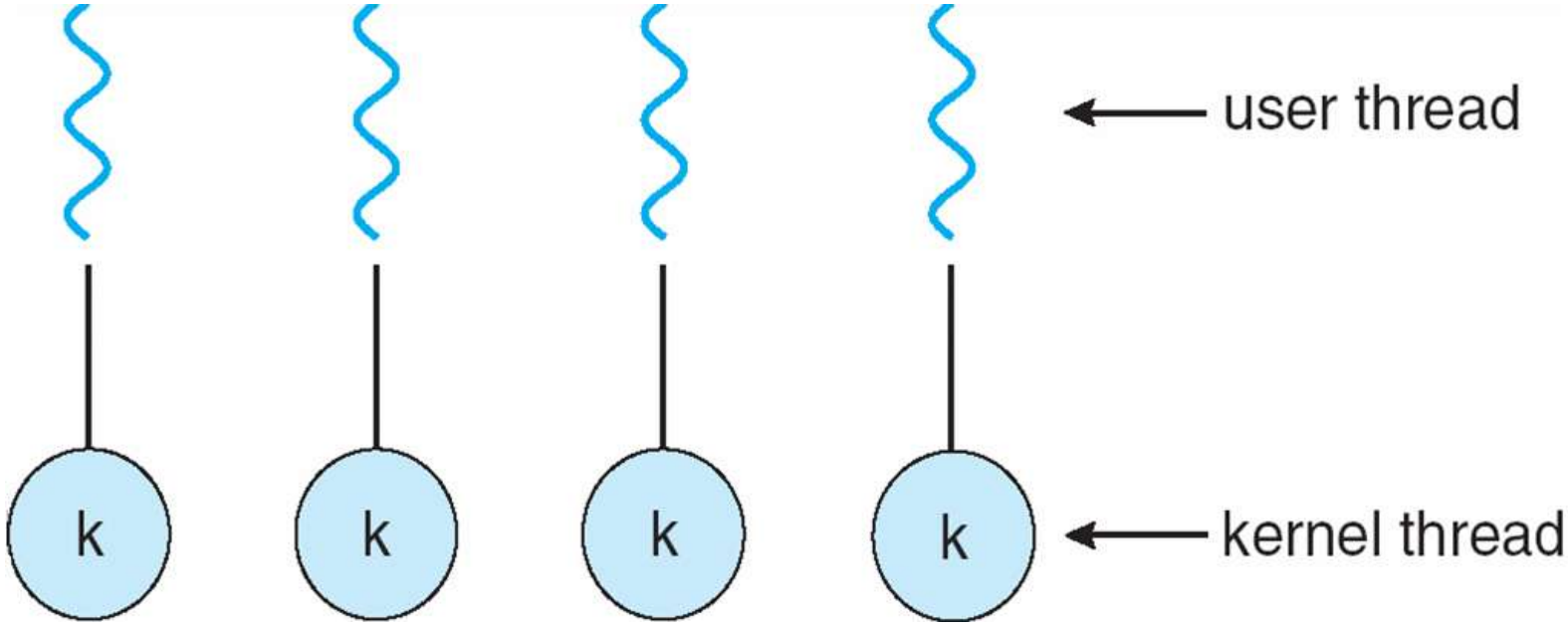
Many-to-One Model

- In this model, many user level threads are mapped to one kernel level thread.
- Threads are managed in user space.



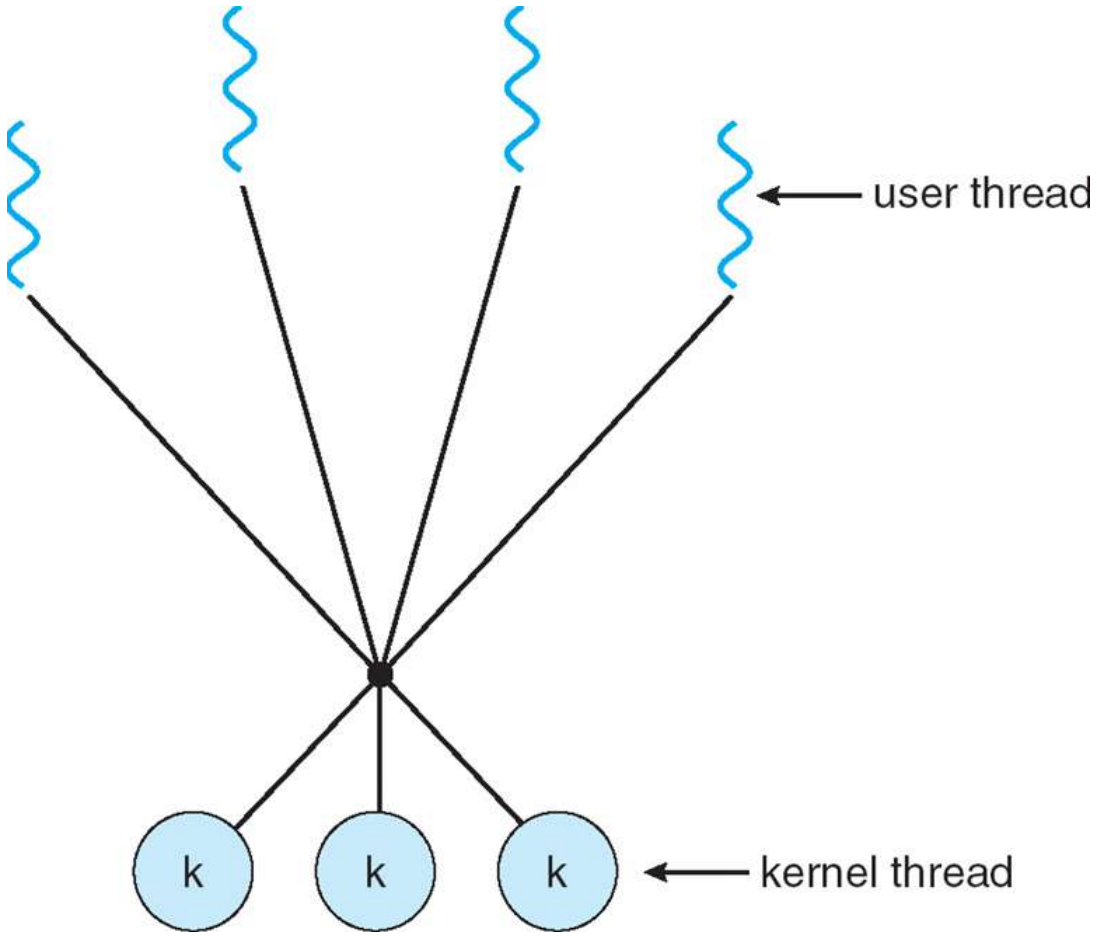
One-to-One Model

- In this model, each user level thread is mapped to one kernel level thread.



MANY-TO-MANY MODEL

- In this model, many user level threads are mapped to many kernel level threads.



Thank You

Have a Nice Day