SNS COLLEGE OF TECHNOLOGY

Kurumbapalayam (Po), Coimbatore – 641 107

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

COURSE NAME : **23CST202-OPERATING SYSTEMS**

II YEAR / IV SEMESTER

Unit 1-OVERVIEW AND PROCESS MANAGEMENT

Topic :Threads

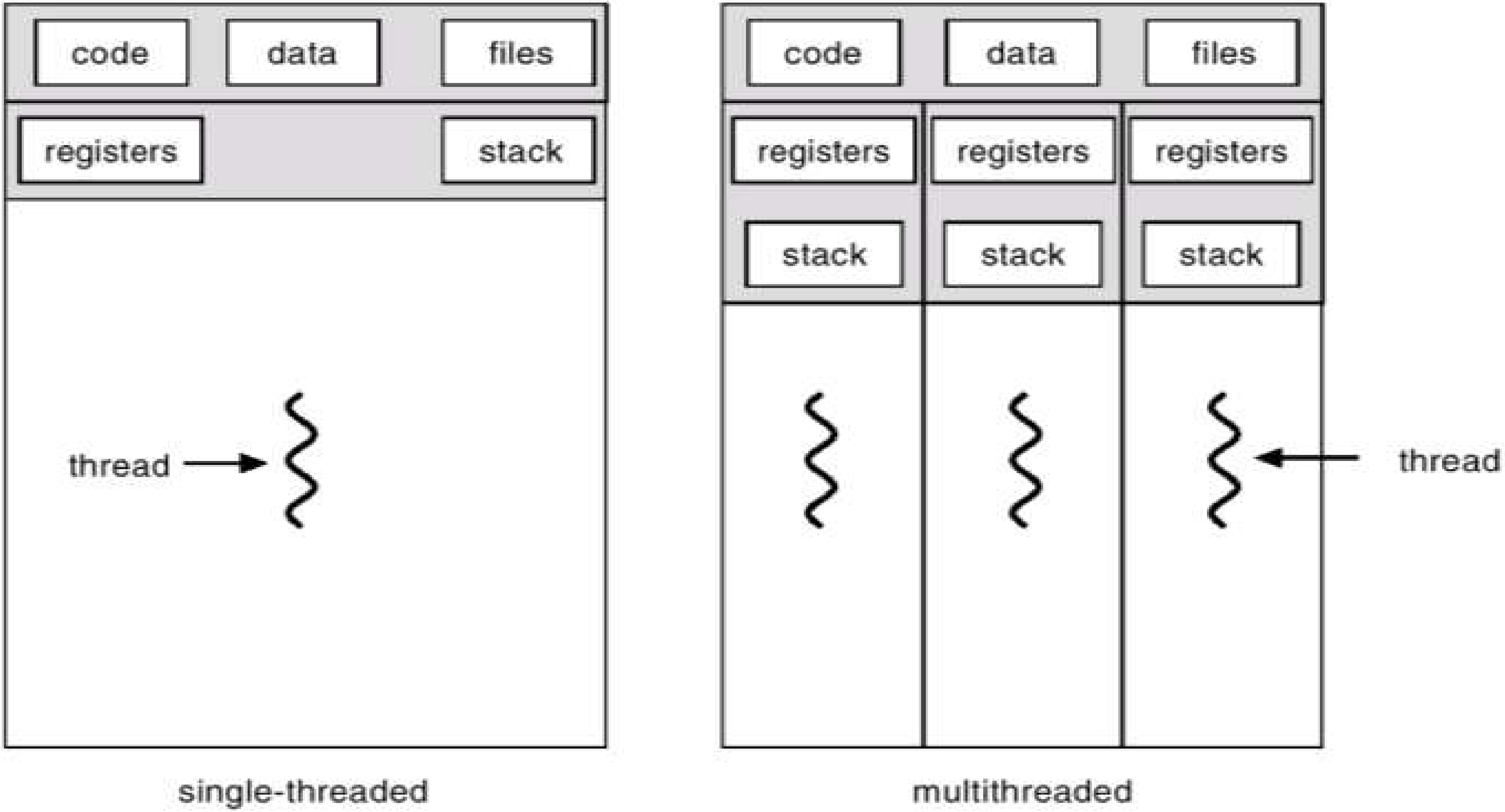
THREADS

Process with a single thread of execution is called Heavy weight process

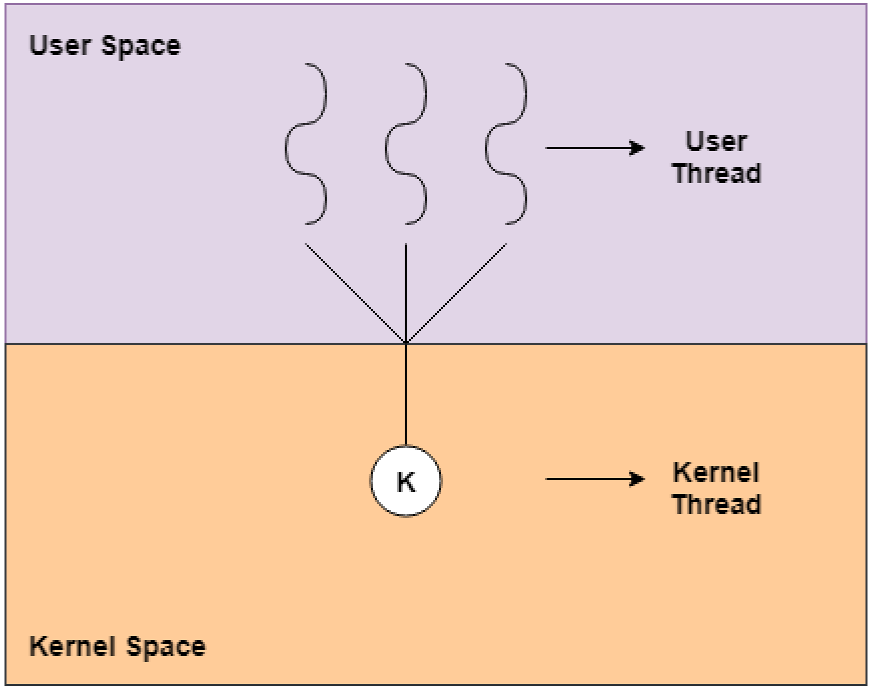
- Thread =light weight process

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

Single and Multithreaded Processes



User Thread and Kernal Thread



User Thread

Thread implementation & management done by user-level threads library (thread _create, thread_exit, thread_wait, etc.) rather than via system calls.

■User thread use →thread switching (when a thread voluntarily goes to sleep – without making any system call) does not need to call the O.S. & to cause an interrupt to the kernel. So switching between user level thread can be done independently of the O.S. & there for very quickly .

■Disadvantages:

■→If the kernel is single threaded, then any user level thread executing a system call will cause the entire task to wait, until the system call returns because kernel schedule only processes & processes waiting for I/O (system call) are put in wait queue & can not be allotted CPU.

■Unfair Scheduling →A process containing single thread say t1 will get 100 times more chances to run than a thread t2 which is one of the threads in process p2 containing 100 threads.

■Examples (user level thread libraries)

-POSIX *Pthreads,* Mach *C-threads,* Solaris *threads*

Kernel Threads

The threads are implemented & managed with the help of O.S. kernel. The smallest unit of processing the kernel will recognize & thus schedule is a thread. →So each thread may be schedule independently.

■→So process B could receive 100 times the CPU time than process A receives.

■Now if a thread make an I/O and a system call, the whole process need not be blocked (only that thread is blocked) & thus another thread in the same process run during this time.

■Examples
-Windows 95/98/NT/2000
-Solaris
-Tru64 UNIX
-BeOS
-Linux

Multithreading Models

There can be two types of kernel also

→Single threaded

→Multi threaded

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

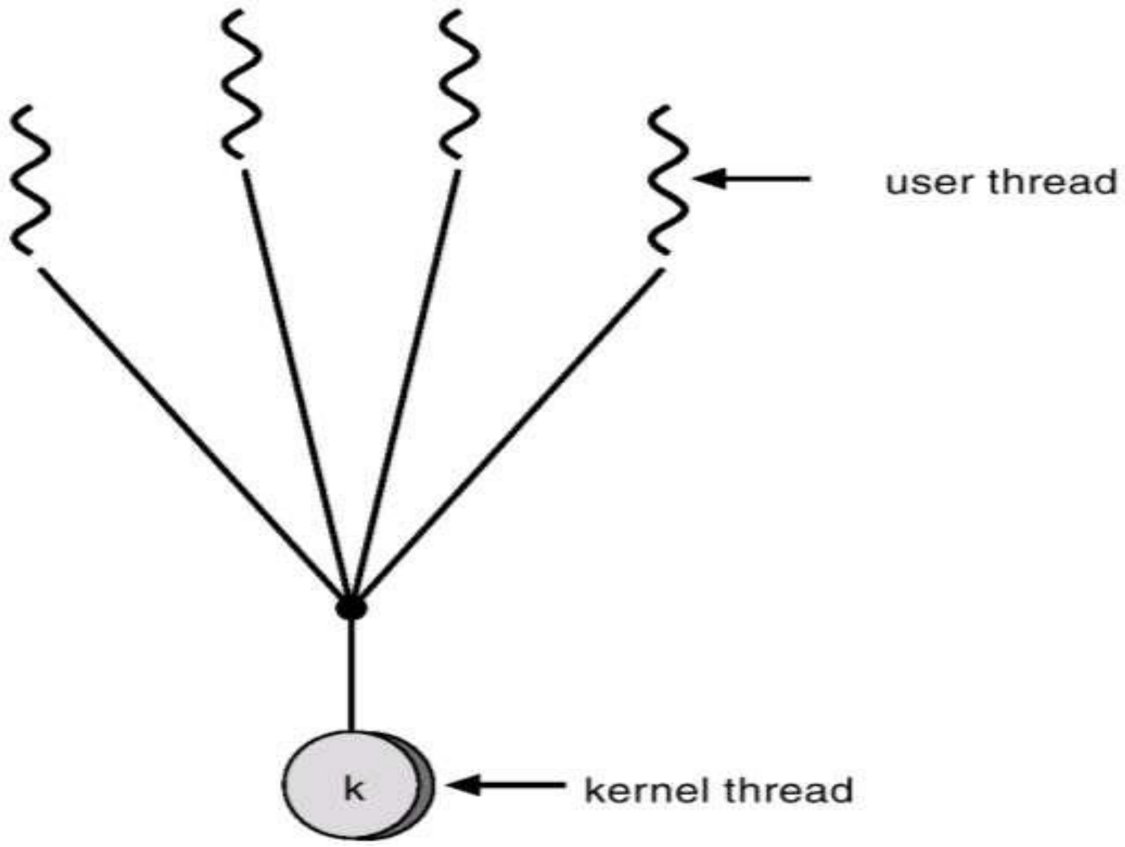
Many - one

Many user-level threads mapped to single kernel thread.

■Used on systems that do not support kernel threads.

→Example :-Unix

Many-to-One Model



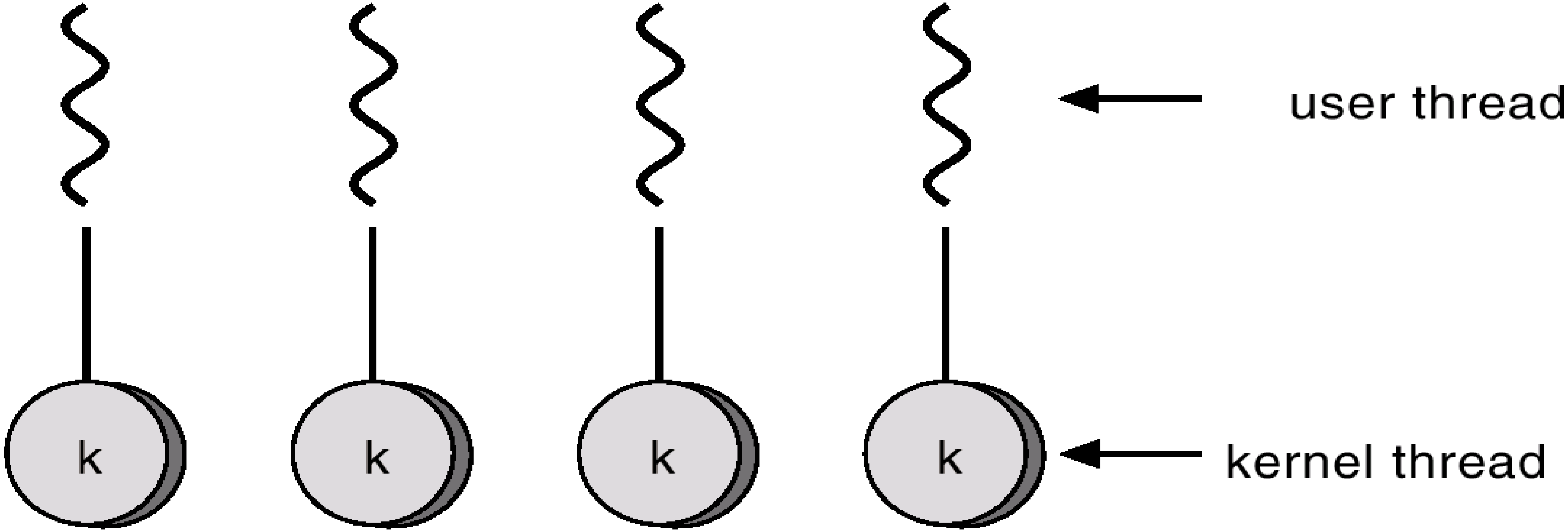
One-to-One Model

Each user-level thread maps to kernel thread.

■Examples

-Windows 95/98/NT/2000

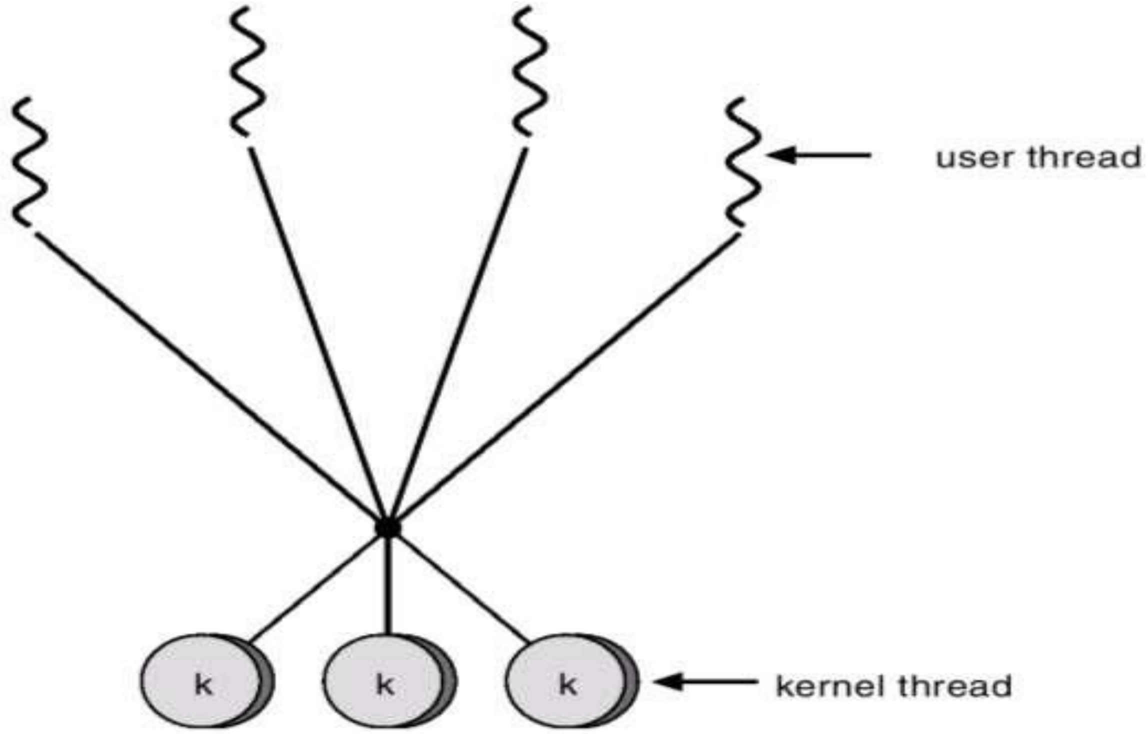
-OS/2



Many-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 of target thread

Asynchronous or deferred

Signal handling

Thread pools

Thread-specific data

Scheduler activations

Thread Cancellation

Terminating a thread before it has finished

Two general approaches:

Asynchronous cancellation terminates the target thread immediately

Deferred cancellation allows the target thread to periodically check if it should be cancelled

Signal Handling

Signals are used in UNIX systems to notify a process that a particular event has occurred

A signal handler is used to process signals

1. Signal is generated by particular event
2. Signal is delivered to a process
3. Signal is handled

Thread Pools

Create a number of threads in a pool where they await work

Advantages:

Usually slightly faster to service a request with an existing thread than create a new thread

Allows the number of threads in the application(s) to be bound to the size of the pool

Thread Specific Data

Allows each thread to have its own copy of data

Useful when you do not have control over the thread creation process

(i.e., when using a thread pool)

Scheduler Activations

Scheduler activations provide upcalls - a communication mechanism

from the kernel to the thread library

This communication allows an application to maintain the correct

number kernel threads

THANK YOU