Operating Systems
CS F372

Threads

BIJU K RAVEENDRAN

Threads

- 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

Threads

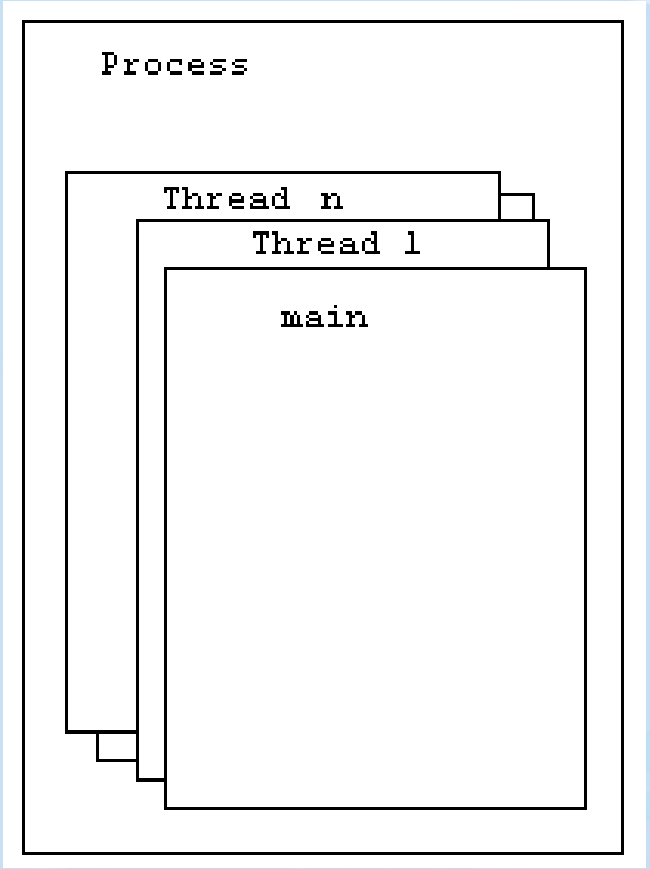
- A *thread* (or *lightweight process*) is a basic unit of CPU utilization; it consists of:
  - Thread ID
  - program counter
  - register set, stack pointer
  - stack space for local variables and return addresses
  - Signal mask
  - Priority
  - Return value : errno

Threads

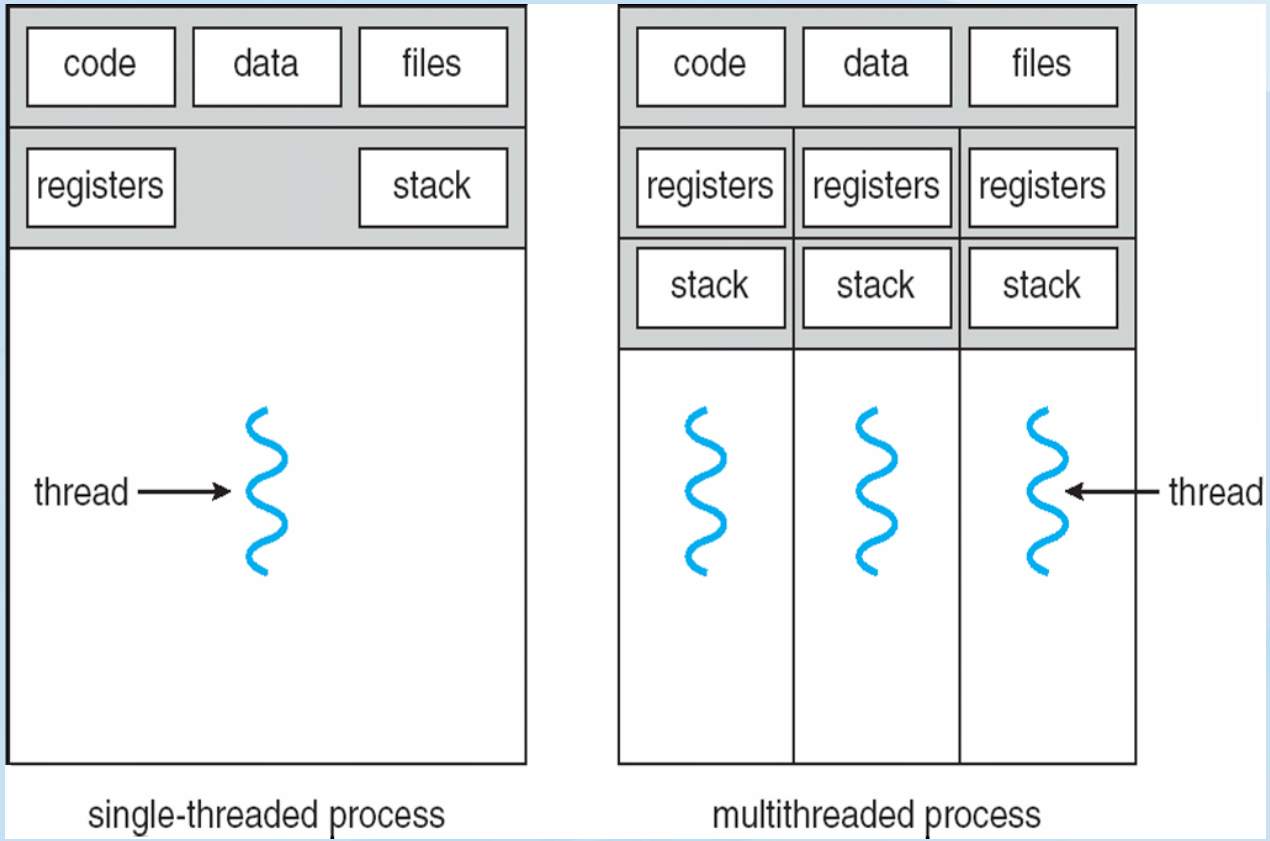
- A thread shares with its peer threads its:
  - Code section
  - Global data section
  - Operating-system resources
  - Process instructions
  - Open files (descriptors)
  - Signal and signal handlers
  - Current working directory
  - User and group id

  And is collectively know as a *task*.
- A traditional or *heavyweight* process is equal to a task with one thread
- Unlike process thread does not maintain a list of created threads.
- It does not know the thread created it also

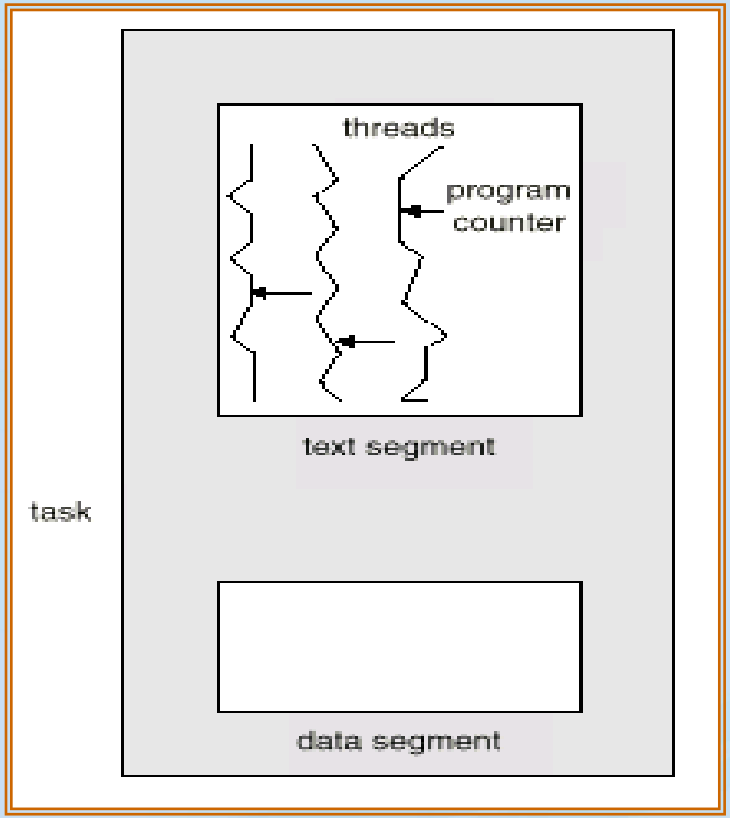
Process & Threads



Single and Multi-threaded Processes



Multiple threads within a Task



Thread

Local to a thread

- Thread ID
- program counter
- register set, stack pointer
- stack space for local variables
- return addresses
- Signal mask
- Priority
- Return value : errno

Global to all threads

- Code section
- Global data section
- Operating-system resources
- Process instructions
- Open files (descriptors)
- Signal and signal handlers
- Current working directory
- User and group id

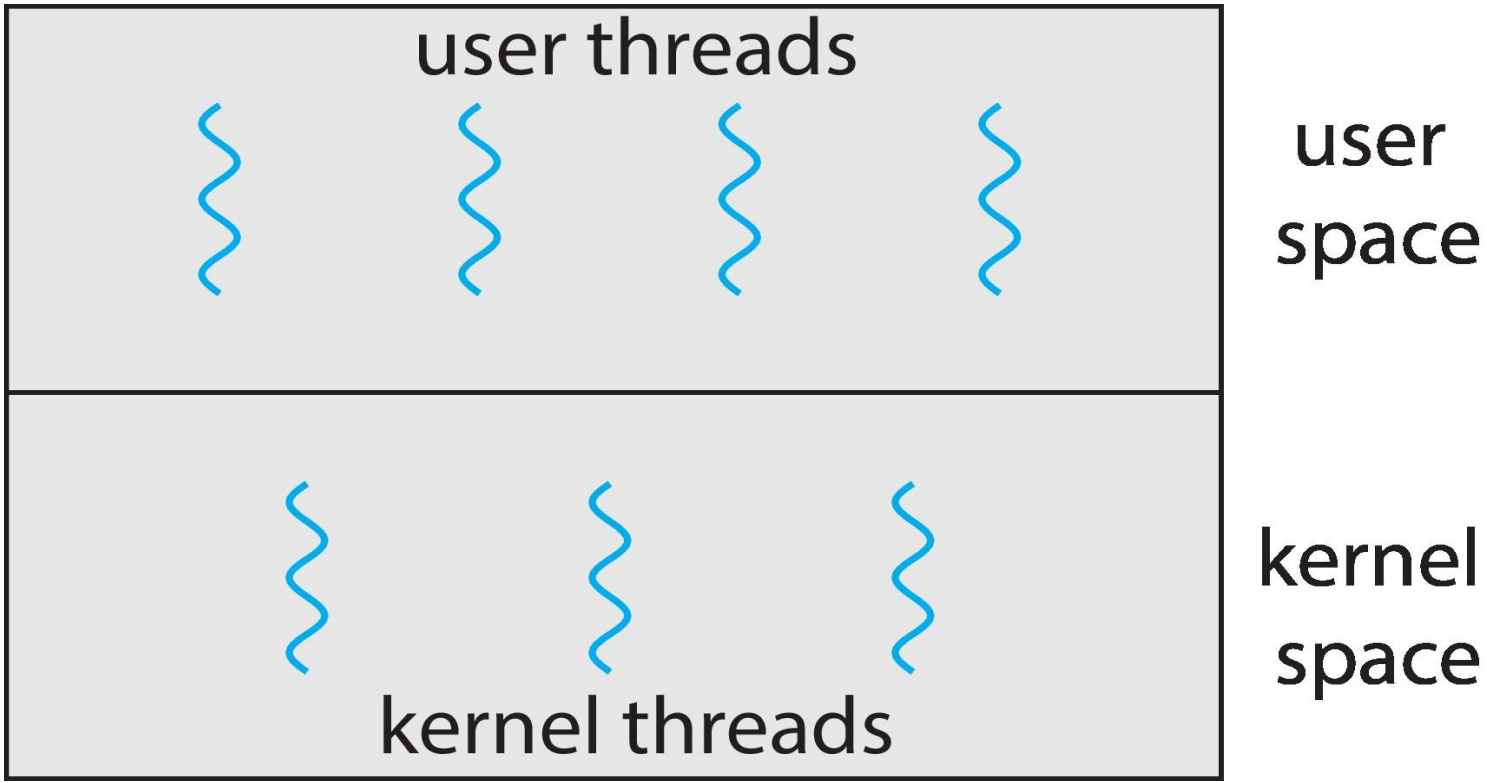
User Threads

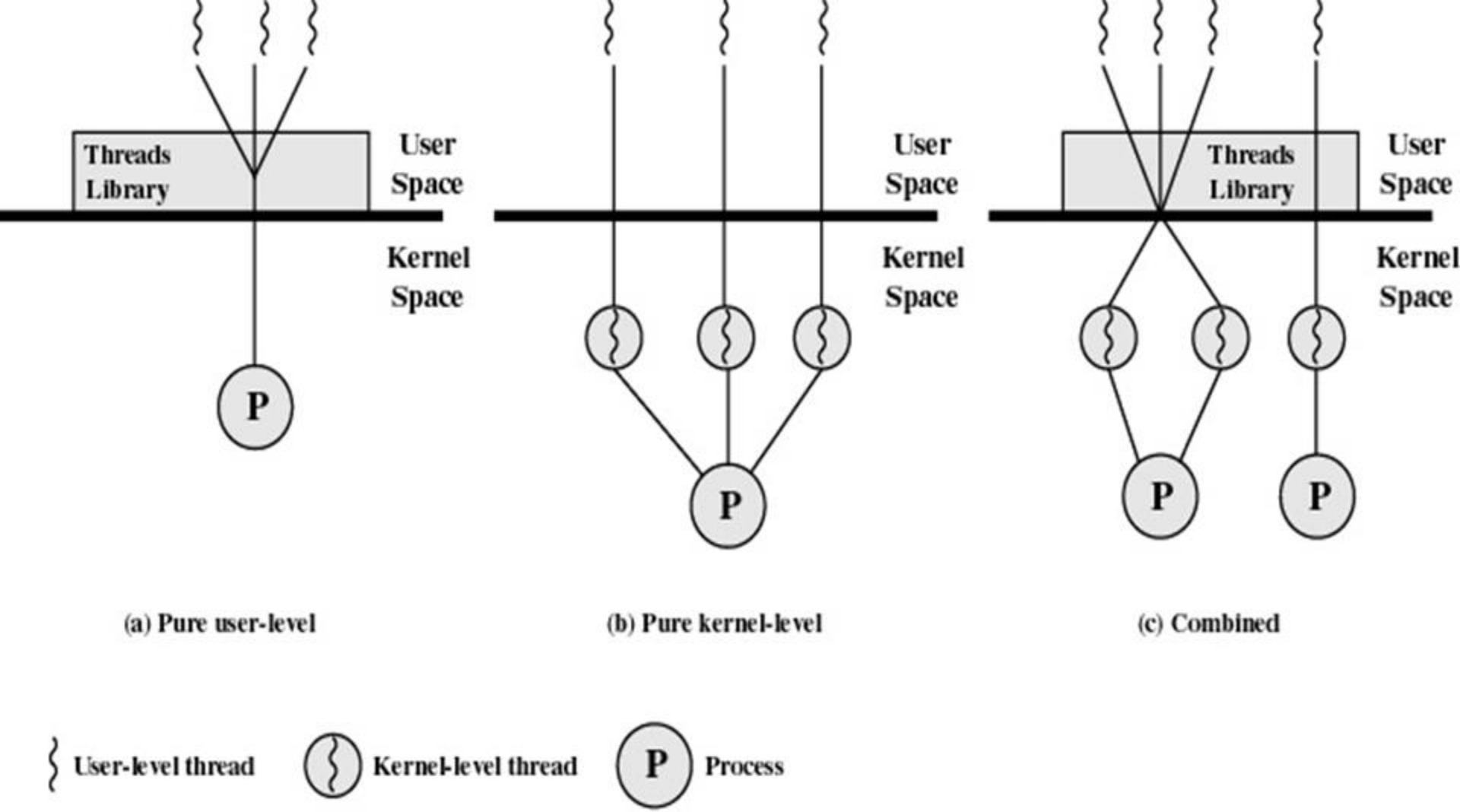
- These threads are supported above the kernel
- Implemented by thread library at the user level
- All thread management is done by the application
- The kernel is not aware of the existence of threads
- Fast to create and manage
- Thread management done by user-level threads library
- Three primary thread libraries:
  - POSIX Pthreads, Java threads, Win32 threads

Kernel Threads

- Supported directly by OS
- Slow compared to user threads
- Scheduling is done by thread basis
- Examples
  - Windows, Solaris, Linux, Mac OS X, iOS, Android

User and Kernel Threads



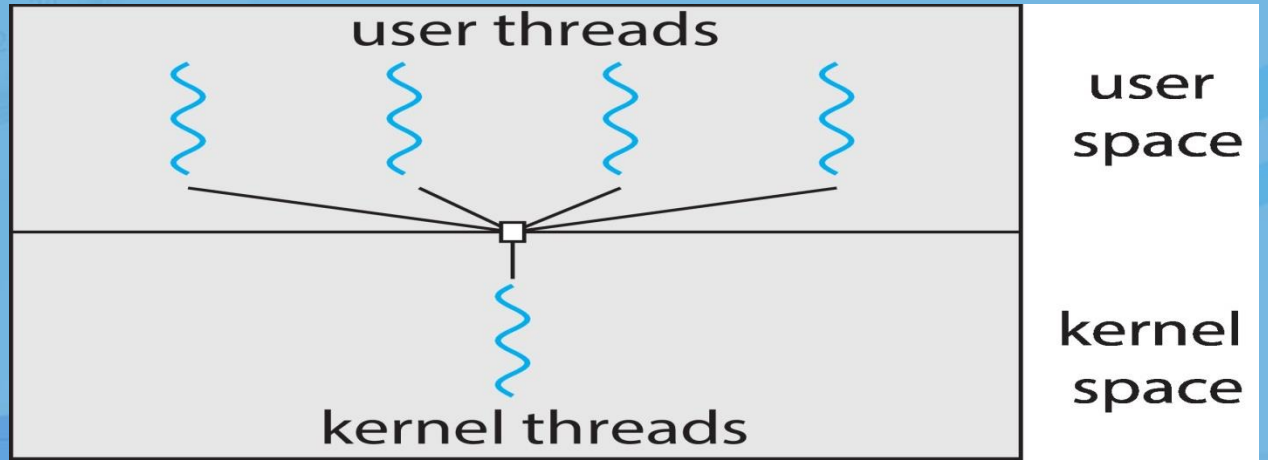

Threads Library
User Space
Kernel Space
P
(a) Pure user-level
User Space
Kernel Space
P
(b) Pure kernel-level
Threads Library
User Space
Kernel Space
P
P
(c) Combined
User-level thread
Kernel-level thread
P
Process

Multi-threading Model

- Many-to-One
- One-to-One
- Many-to-Many
- Two –level

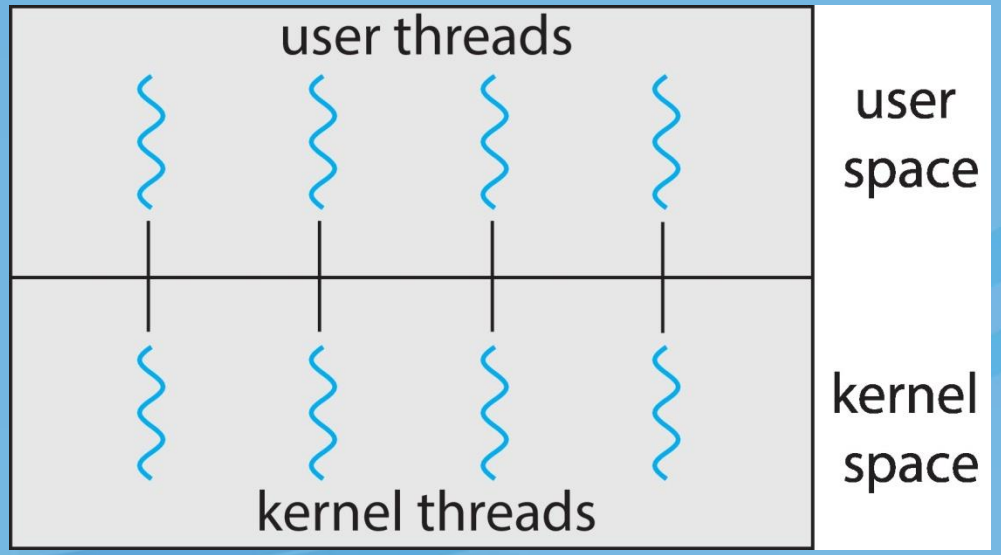
Many – to - One

- 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



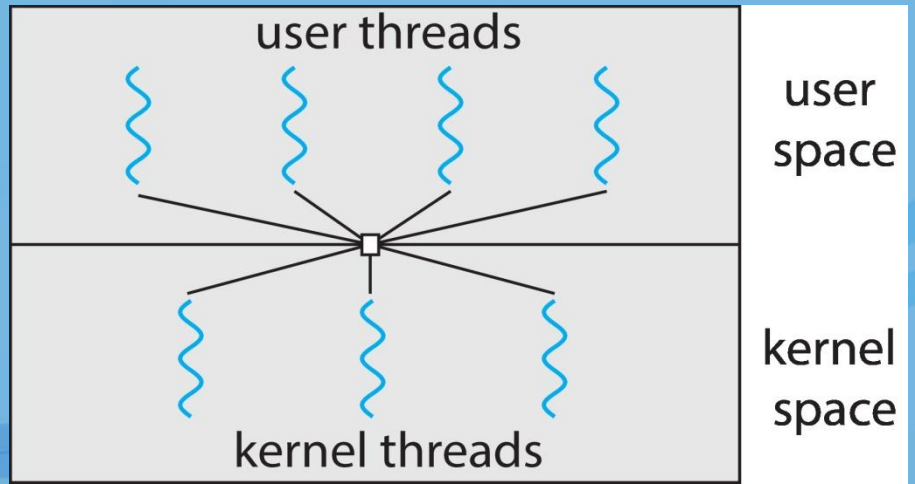
One – to – One Model

- 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



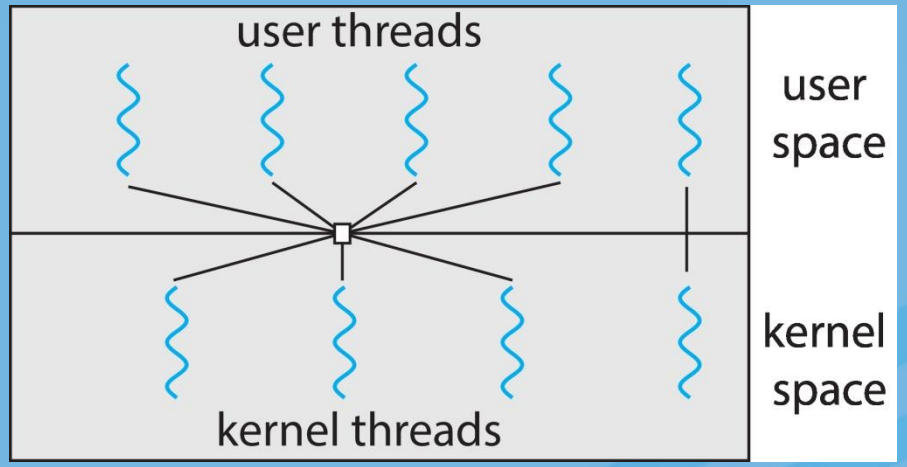
Many – to – Many Model

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



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



Relationship between Threads & Processes

Threads:Process	Description	Example Systems
1:1	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementation
M:1	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Solaris, OS/2, OS/390, MACH

Relationship between Threads & Processes

Threads:Process	Description	Example Systems
1:M	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
M:M	Combines attributes of M:1 and 1:M cases	TRIX