CS343 - Operating Systems

Module-2F

Thread Libraries

Dr. John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

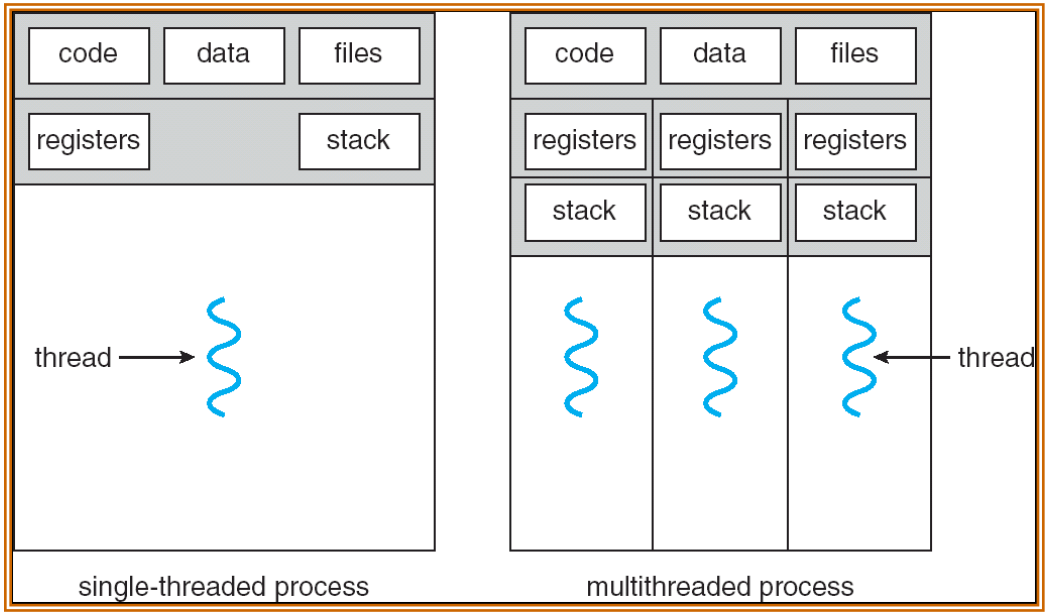
http://www.iitg.ac.in/johnjose/

Session Outline

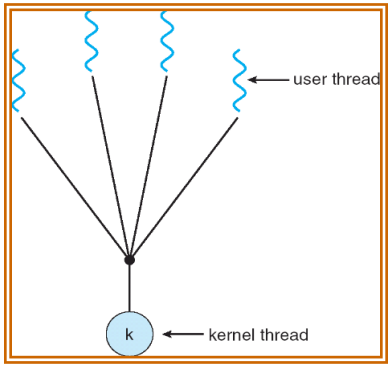
- Review of Thread model
- Light weight Process
- Thread libraries
- Semantics of fork() and exec() system calls
- Thread cancellation
- Signal handling
- Thread pools
- Scheduler activations

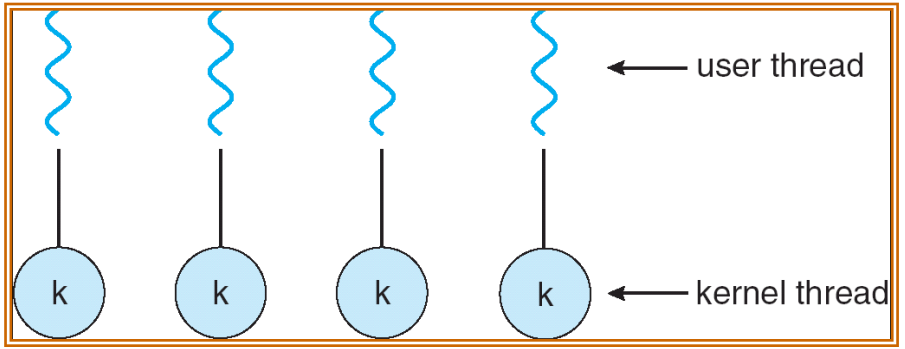
Concept of Threads

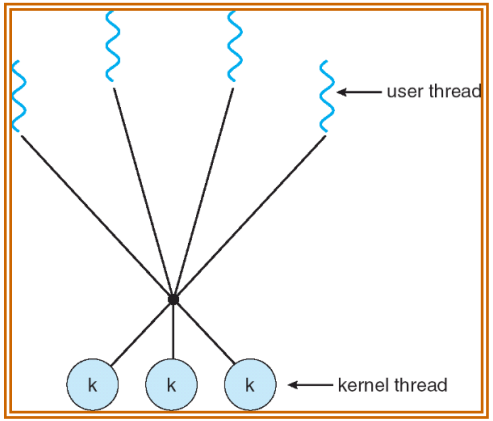
- Thread is a flow of control within a process.

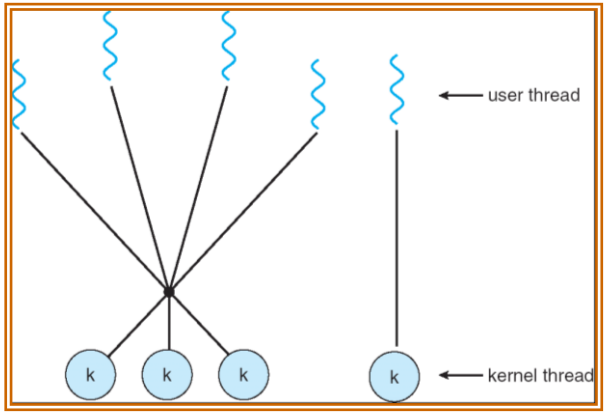


Thread Mappings Models









Thread Libraries

- A **thread library** provides the programmer an API for creating and managing threads
- Two primary ways to implement
  - to provide a library entirely in user space with no kernel support.
    - All code and data structures for the library exist in user space
    - Every function call executes in user mode, not in kernel mode
  - to implement a kernel-level library supported by OS
    - All code and data structures exist in kernel space
    - Invoking functions result in a system call to the kernel

Thread Libraries

- A **thread library** provides the programmer an API for creating and managing threads
- **Three** main thread libraries
  - **POSIX Pthreads** - Solaris, Linux, Mac OS, Tru64 UNIX
  - **Win32 Thread** - Windows
  - **Java Thread** - Java

Pthreads

- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization
- API specifies behavior of the thread library
- Implementation is up to development of the library
- Common in UNIX operating systems (Solaris, Linux, Mac OS X)

Windows XP Threads

- Implements the one-to-one mapping
- Each thread contains a thread context that consists of
  - A thread id
  - Register set
  - Separate user and kernel stacks
  - Private data storage area

Linux Threads

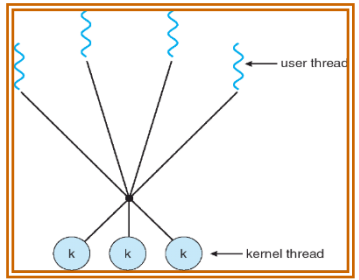
- Linux refers to tasks rather than threads
- Thread creation is done through **clone()** system call
- **clone()** allows a child task to share the address space of the parent task

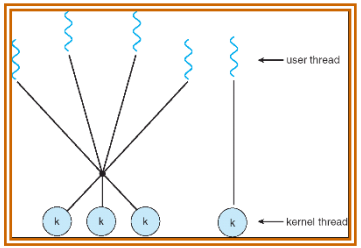
Java Threads

- Java threads are managed by the JVM
- Java threads may be created by:
  - Extending Thread class
  - Implementing the Runnable interface
- Because JVM is running on top of a host OS, the Java thread API is implemented using a thread library available on the host system.
  - On Windows system: using Win32 thread library
  - On Linux and Unix system: using Pthread library

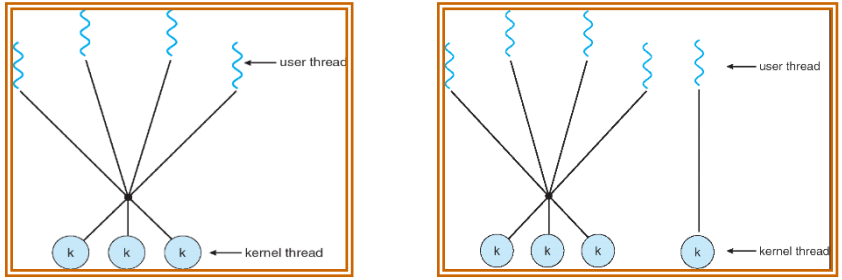
Light Weight Processes (LWP)

- Most popular mapping model - many-to-many or two-level mode
- Light Weight Process (LWP)
  - An intermediate data structure between user and kernel threads
  - A user thread is attached to a LWP
  - Each LWP is attached to a kernel thread
  - OS schedules the kernel threads (not processes) to run on CPU
  - If a kernel thread blocks, LWP blocks, and the user thread blocks





Light Weight Processes (LWP)



- To the user thread library, LWP appears to be a virtual CPU on which the application can schedule a user thread to run.
- The user thread library is responsible for scheduling among user threads to the LWPs. It is similar to the CPU scheduling in kernel.
- In general, context switching between user threads involves taking a user thread of its LWP and replacing it with another thread.

Threading Issues

- Semantics of fork() and exec() system calls
- Thread cancellation
- Signal handling
- Thread pools
- Scheduler activations

Semantics of fork() and exec()

- **fork()** - system call is used to create a duplicate process.
- **exec()** - system call is used to create a new separate process.
- Fork() starts a new process which is a copy of the one that calls it, while exec replaces the current process image with another new one.
- Both parent and child processes are executed simultaneously in case of fork()
- In exec() control never returns to the original program unless there is an exec() error.

Thread Cancellation

- Terminating a thread before it has finished by other threads
  - Ex, multiple threads search DB, one thread returns result. The remaining thread might be canceled.
- Two general approaches to cancel the target thread
  - **Asynchronous cancellation** terminates the target thread immediately
  - **Deferred cancellation** allows the target thread to periodically check if it should be cancelled

Thread Cancellation

- **Asynchronous cancellation**
  - The difficulty with cancellation occurs in situation where
    - resources have been allocated to a canceled thread, or
    - where a thread is canceled while in the midst of updating data it is sharing with other threads
- **Deferred cancellation**
  - One thread indicates that target thread is to be canceled.
  - Cancellation occurs only after the target thread has checked a flag to determine if it should be canceled or not

Signal Handling

- **Signals** are used in UNIX systems to notify a process that a particular event has occurred
- **Two types of signals**
  - Synchronous signals [illegal memory access, division by 0]
  - Asynchronous signals [Specific keystrokes (Ctrl-C), timer expire]
- What happen when a signal generated?
  - Signal is generated by particular event
  - Signal is delivered from kernel to a process
  - Signal is handled

Signal Handling

- Every signal may be handled by one of two possible handlers.
  - A default signal handler
  - A user-defined signal handler
- Every signal has a default signal handler that is run by the kernel
- The default action can be overridden by a user-defined signal handler
- Options when a signal occurs on a multi-threaded process
  - Deliver the signal to the thread to which the signal applies
  - Deliver the signal to every thread in the process
  - Deliver the signal to certain threads in the process
  - Assign a specific thread to receive all signals for the process

Thread Pools

- **Pool** of threads where they await work
- A process creates few threads at start up and place into a pool
- When receiving a request, server awakens a thread from the pool and passes the request to service
- Ones the thread completes its service, it returns to the pool and await more work.
- If the pool contains no available thread, the server waits until one becomes free.
- Thread pools are 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

Scheduler Activations

- Both Many-to-Many and Two-level models require communication to maintain the appropriate number of kernel threads allocated to the application
- 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

Summary

- A thread is a flow of control within process.
- Multithreaded process contains several different flows of control within the same address space.
- Benefits of multi-threading includes
  - Increased responsiveness, Resource sharing within the process
  - Economy, Ability to take advantage of multiprocessor architecture
- User-level thread are thread that are visible to the programmer and are unknown to the kernel.

Summary

- OS kernel supports and manages kernel-level threads
- Three types of models relates user and kernel threads
  - One-to-one, many-to-one, many-to-many
- Thread libraries provide the application programmer with an API for creating and managing threads
  - POSIX Pthreads, Win32 threads, Java threads
- Multithreaded programs introduces several issues
  - fork()/exec(), thread cancellation, signal handling, thread pools and schedule activation.

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/