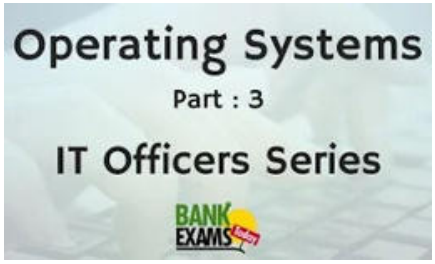
Operating Systems - Part 3

Published on Tuesday, February 16, 2016



Hi Folks,
Hope your IBPS IT officer's exams went well. A lot of professional section questions were from PK bundle and articles shared here. All the best for results!

We'll be continuing with operating systems and start with 'Threads'. Previous article on OS can be found at operating systems - part 2.

Threads

A thread is a basic unit of CPU utilization, consisting of a program counter, a stack and set of registers. It is also called a **light weight process**. Each thread belongs a process and can not exist outside a process. They allow us to **parallely execute** application on shared multiprocessors.

Like processes, threads share CPU and can create child threads using fork ( ) system call. All threads can access every address in the task.

What's so good about them?

Multiple threads can **share common data** thereby eliminating the need for inter process communication. Also they consume very **less resources** as compared to processes. **Context switching** is more efficient with threads. We'll discuss about context switching in later articles.

Types of Threads

1. User level threads

- Implement in user level libraries **not via system calls**.
- No modification to operating system is required.
- Creation of thread, switching & sync between them can be done **without interrupting the kernel**.

2. Kernel level threads

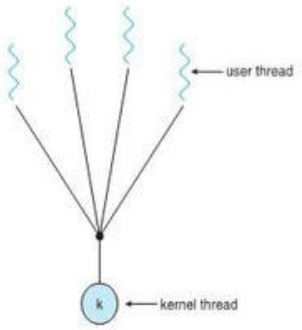
- Kernel manages the threads by keeping track of all in a **thread table**.
- Uses **system calls** to create and manage threads.
- Scheduler may allocate more time to a process having more threads since kernel knows about them all.
- Much **slower** than user level threads.

Multi threading mo dels

1. Many to One

Basic features of this model are as follows:

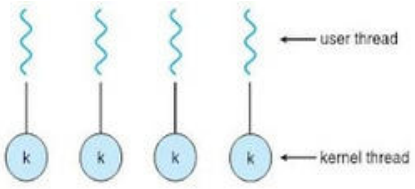
- Many user level threads mapped to a single kernel thread.
- Blocking of one thread causes all to block.
- Multiple threads may not run in parallel as only one in kernel at a time.
- E.g. - Solaris Green Threads, GNU Portable Threads.



2. One to One

Basic features are as follows:

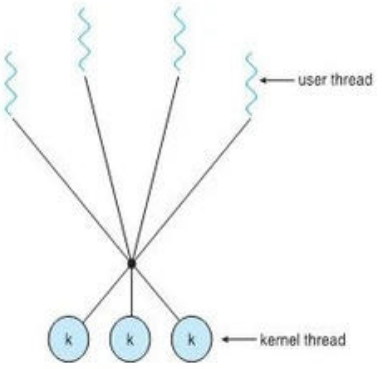
- Each user level thread is mapped to kernel thread.
- If you create a user thread, a kernel thread will be created.
- Number of threads per process are sometimes restricted.
- E.g. - Linux, Windows.



3. Many to Many

Basic features are as follows:

- Many user level threads are mapped to many kernel level threads.
- If kernel system calls are blocked, it doesn't block the entire process.
- Users have no restrictions on number of threads created.



Thread Libraries

A thread library provides an API to developer for creating and managing threads.
There are 3 main thread libraries that are used across operating systems:

- POSIX Pthreads
- Windows threads
-

Java threads

Issues related to Threads

1. Fork ( ) system call

The basic issue that arises is when a thread which forks ( ), duplicates itself only or all other threads which are associated with the process. Unix systems handle this by having two versions of fork ( ) system calls and an option suited to requirement is chosen.

2. Signal Handling

Signals in OS are used to notify a process that some event has occurred. A signal handler processes all signals. Every signal has a default handler assigned by kernel. User defined signals can over ride them.

3. Thread Cancellation

Threads can be terminated before they have finished in case they are not required anymore. Two approaches are used for cancelling threads:

- **Asynchronous cancellation** - it terminates the thread immediately.
- **Deferred cancellation** - thread is periodically checked if it should be cancelled.

On Linux systems, thread cancellation is handled through signals.

Interesting fact about OS

NASA's launch countdown clock runs on Fedora Servers.