Mawlana Bhashani Science and Technology University

Lab-Report

Report No:03

Course code:ICT-3110

Course title:Operating Systems Lab

Date of Performance: 03-09-2020

Date of Submission: 16-09-2020

Submitted by

Name:Meraz Ahmed

ID:IT-18005

3rd year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment No: 03

Experiment Name: Threads on operating system

Q1:What is Thread?

Ans:A thread is a path of execution within a process. A process can contain multiple threads.A thread is the smallest unit of processing that can be performed in an OS. In most modern operating systems, a thread exists within a process - that is, a single process may contain multiple threads.

Q2:WriteTypes of Threads ?

Ans : There are three types of threads in operating system. They are –

1. User level thread
2. Kernel level thread
3. **User Level thread :**
   Is implemented in the user level library, they are not created using the system calls. Thread switching does not need to call OS and to cause interrupt to Kernel. Kernel doesn't know about the user level thread and manages them as if they were single-threaded processes.

Advantages of ULT:

a. Can be implemented on an OS that does't support multithreading.
b. Simple representation since thread has only program counter, register set, stack space.
c. Simple to create since no intervention of kernel.
d. Thread switching is fast since no OS calls need to be made.

Disadvantages of ULT :

e. No or less co-ordination among the threads and Kernel.
f. If one thread causes a page fault, the entire process blocks.

4. **Kernel Level Thread :**
   Kernel knows and manages the threads. Instead of thread table in each process, the kernel itself has thread table (a master one) that keeps track of all the threads in the system. In addition kernel also maintains the traditional process table to keep track of the processes. OS kernel provides system call to create and manage threads.

Advantages of KLT:

a. Since kernel has full knowledge about the threads in the system, scheduler may decide to give more time to processes having large number of threads.
b. Good for applications that frequently block.

Disadvantages of KLT:

c. Slow and inefficient.
d. It requires thread control block so it is an overhead.

Q3 :Write the implementation of threads.

Ans:

Threads Implementation in User Space:

In this model of implementing the threads package completely in user space, the kernel don't know anything about them.

The advantage of implementing threads package in user space is that a user-level threads package can be implemented on an OS (Operating System) that doesn't support threads.

Threads Implementation in Kernel:

In this method of implementing the threads package entirely in the kernel, no any run-time system is need in each as illustrated in the figure given below.

In this, there is no any thread table in each process. But to keep track of all the threads in the system, the kernel has the thread table.

Whenever a thread wants to create a new thread or destroy an existing thread, then it makes a kernel call, which does the creation or destruction just by updating the kernel thread table.

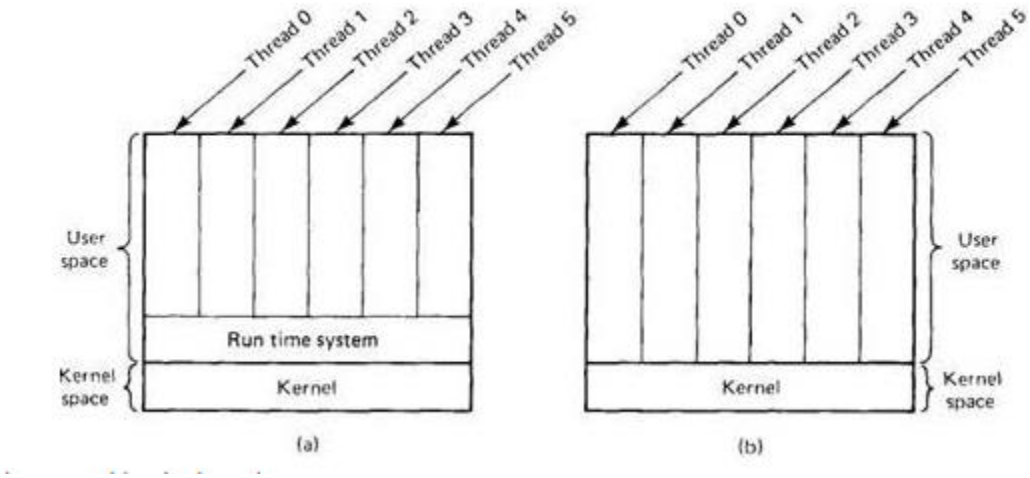
The thread table of the kernel holds each registers, state, and some other useful information of the thread.

Here the information is the same as with the user-level threads.

This information is a subset of the information that traditional kernels maintains about each of their single-threaded processes, that is, the process state.

In addition to these, to keep track of processes, the kernel also maintains the traditional process table. Another, and probably most devastating argument against user-level threads is that programmers generally want threads in applications where the threads block often, as, for example, in a multithreaded file server. These threads are constantly making system calls. Once a trap has occurred to the kernel. to carry out the system call, it is hardly any more work for the kernel to switch threads if the old one has blocked, and having the kernel do this eliminates the need for constantly checking to see if system calls are

safe. For applications that are essentially entirely CPU bound and rarely block, what is the point of having threads at all? No one would seriously propose to compute the first n prime numbers or play chess using threads because there is nothing to be gained by doing it that way.



(a) A user-level threads package. (b) A threads packaged managed by the kernel.