Threads

Reading: Section 2.2 of the Textbook (Tanenbaum)

What is Thread?

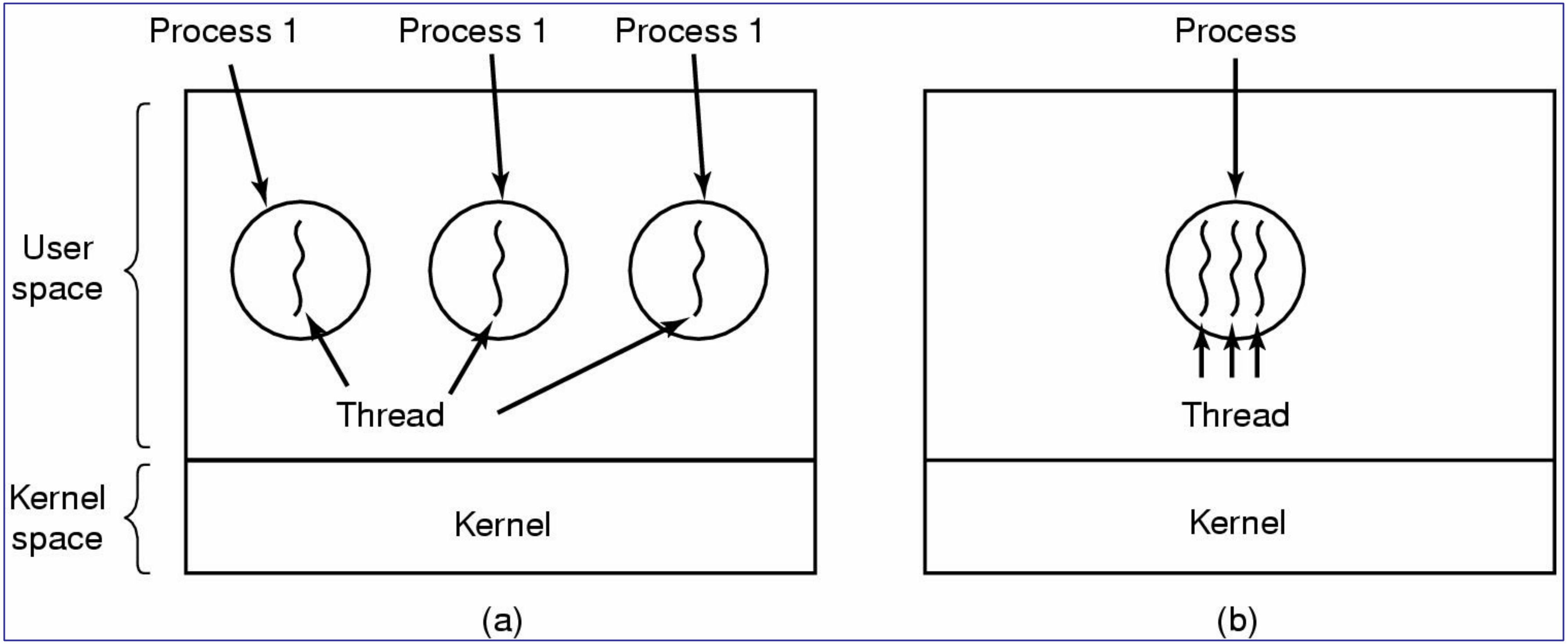
Threads, like process, are a mechanism to allow a program to do more than one thing at a time.

Conceptually, a thread (also called *lightweight process*) exists within a process (heavyweight process). Threads are a finer-grained unit of execution than processes

Threads

- Traditional threads has its own address space and a single thread of control.
- The term multithreading is used to describe the situation of allowing the multiple threads in same process.
- When multithreaded process is run on a single-CPU system, the threads take turns running as in the multiple processes.
- All threads share the same address space, global variables, set of open file, child processes, alarms and signals etc.

Threads

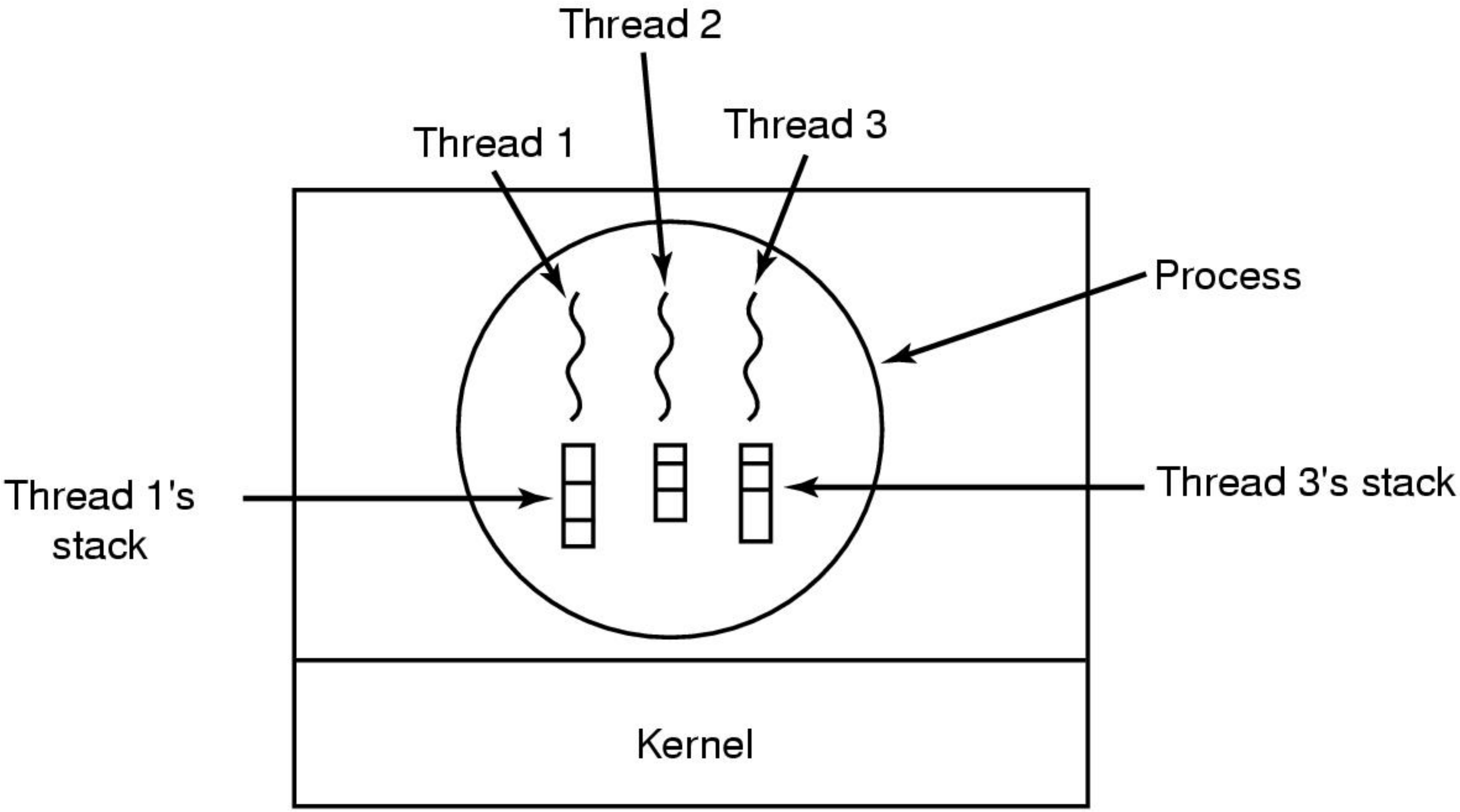


(a) Three process each with one thread (b) one process with three threads.

Threads

- *Figure (a)* organization is used when three processes are unrelated whereas *(b)* would be appropriate when the three threads are actually part of the same job and are actively and closely cooperating with each other.
- Each thread maintain its own stack.

Threads


Thread 2
Thread 1
Thread 3
Process
Thread 1's stack
Thread 3's stack
Kernel

Threads

Example – Word processor

Needs to accept user input, display it on screen, spell check, auto save and grammar check.

- Implicit: Write code that reads user input, displays/formats it on screen, calls spell checked etc. while making sure that interactive response does not suffer.
- Threads: Use threads to perform each task and communicate using queues and shared data structures
- Processes: expensive to create and do not share data structures and so explicitly passed.

Others: Spreadsheet, Server for www, browser etc.

Threads

Advantages:

- Responsiveness.
- Resource sharing.
- Economy.

Problems: designing complexity.

In Unix system when fork create it copy all threads of parent to the child.

What happens if the thread of parent blocked ?

When the line typed, do both thread get a copy of it?

Threads share the many data structure.

What happens if one thread closes a file while another is still reading from it?

Need complex scheduling operations

Users and Kernel Threads

User Threads:

Thread management done by user-level threads library.

- Implemented as a library
- Library provides support for thread creation, scheduling and management with no support from the kernel.
- Fast to create
- If kernel is sigle threaded, blocking system calls will cause the entire process to block.
- Example: POSIX Pthreads, Mach C-threads.

Kernel Threads:

Supported by the Kernel

- Kernel performs thread creation, scheduling and management in kernel space.
- Slower to create and manage

1. Blocking system calls are no problem
2. Most OS’s support these threads
3. Examples: WinX, Linux

Home works

HW #3:

1. Q. 7, 8, 9 & 12 from the Textbook (Tanenbaum)

2. Discribe how multithreading improve performance over a singled-threaded solution.

3.What are the two differences between the kernel level threads and user level threads? Which one has a better performance?

4. List the differences between processes and threads.

5. What resources are used when a thread is created? How do they differ from those used when a process is created?

Reading: Section 2.3 of Textbook (Tanenbaum)