Tutorial - Chapter 3

1. Discuss the different states that a process can exist in at any given time.

   Answer
   The possible states of a process are: new, running, waiting, ready, and terminated. The process is created while it is in the new state. In the running or waiting state, the process is executing or waiting for an event to occur, respectively. The ready state occurs when the process is ready and waiting to be assigned to a processor and should not be confused with the waiting state mentioned earlier. After the process is finished executing its code, it enters the termination state.

2. Describe the actions taken by a kernel to context-switch between processes.

   Answer
   In general, the operating system must save the state of the currently running process and restore the state of the process scheduled to be run next. Saving the state of a process typically includes the values of all the CPU registers in addition to memory allocation. Context switches must also perform many architecture-specific operations, including flushing data and instruction caches.

3. What are the differences between a short-term and long-term scheduler?

   Answer
   The primary distinction between the two schedulers lies in the frequency of execution. The short-term scheduler is designed to frequently select a new process for the CPU, at least once every 100 milliseconds. Because of the short time between executions, the short-term scheduler must be fast. The long-term scheduler executes much less frequently; minutes may separate the creation of one new process and the next. The long-term scheduler controls the degree of multiprogramming. Because of the longer interval between executions, the long-term scheduler can afford to take more time to decide which process should be selected for execution.

4. Discuss in details the producer/consumer problem.

   Answer
   In the producer/consumer problem, a producer creates items and a consumer uses them. The items are stored in a shared buffer, which can be infinite or of a limited size. The producer and consumer must synchronize on the buffer contents so that items are not lost or consumed more than once. If the buffer is empty, the consumer must wait for the producer to create a new item. If a finite buffer is full, the producer must wait for the consumer to use an item.

5. List three different situations lead to the following direct transitions between states: Running state to ready state; waiting state to ready state; running state to waiting state

Answer

- Running state to ready state
  Answer: Interrupt, time-out (end of time slice), arrival of a higher priority process, sleep system call.
- Waiting state to ready state
  Answer: End of I/O, parent becomes ready after child complete execution
- Running state to waiting state
  Answer: Request for I/O, process forks a child, wait for an event, wait system call.

6. Describe the actions taken when a user process starts to execute a write() system call.

Answer

1. The write() call is a privileged instruction that traps to the kernel.
2. The kernel saves the user process context.
3. The kernel sets up registers in the I/O controller and transfers the data to be written to the controller.
4. The kernel selects a (different) ready process and switches context to that process.
5. When output is complete, the I/O device generates an interrupt.
6. The interrupt service routine saves the current CPU context, does some accounting, and moves the writing user process into the ready queue.
7. The kernel either 1) switches context to the original process executing the system call, or 2) returns control to the second process, or 3) selects and switches context to an entirely different process.

Tutorial - Chapter 4

1. Give programming examples in which multithreading provides better performance than a single-threaded solution.

Answer

1. A web server that services each request in a separate thread.
2. A web browser with separate threads playing sound, downloading a file, collecting user input, etc.
3. A word processor with a thread to save, at regular intervals, the file that is currently being executed, and another thread as a spell-checker, etc.
4. An application such as matrix multiplication where each row of the matrix product is evaluated, in parallel, by a different thread.

2. What is the difference between a process and a thread? Which one consumes more resources?

Answer

A process defines the address space, text, resources, etc. A thread defines a single sequential execution stream within a process. Threads are bound to a single process. Each process may have multiple threads of control within it.

It is much easier to communicate between threads than between processes. It is easy for threads to accidentally disrupt each other since they share the entire address space. Process consumes more resources.

3. Context switching between kernel threads typically requires saving some items in its TCB. Mention two such items.

Answer

A process defines the address space, text, resources, etc. A thread defines a single sequential execution stream within a process. Threads are bound to a single process. Each process may have multiple threads of control within it.

4. Assume that the OS implements Many-to-Many multithreading model. What is the minimum number of kernel threads required to achieve better concurrency than in the Many-to-One model? Why?

Answer

At least two kernel threads are required to achieve better concurrency in the many-to many model than in the Many-to-One model. With more than two kernel threads, CPU can still schedule other threads for execution when one threads is performing a blocking system calls.

5. Consider two scenarios: a) two user threads are mapped into one kernel thread, and b) each of the two user threads is mapped into a unique kernel thread. Which achieves better concurrency in execution? Why?

Answer

The b) achieves better concurrency since two threads can execute concurrently.
In a), when one thread is blocked or in waiting state, the other thread cannot be scheduled to run.

6. Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single-processor system?

Answer

A multithreaded system consisting of multiple user-level threads mapped to one kernel thread cannot make use of the different processors in a multiprocessor system. Consequently, there is no performance benefit associated with this solution. The multithreaded solution could be faster if the multiple user-level threads are mapped to different kernel threads.