| | |
|---|---|
| Operating Systems | Wednesday Nov. 14, 2012 |
| CSC 227 | Homework 3 |
| Fall 2012/2013 | Points: 40 |
| Sample Solution | Due:: Sat. 24, 2012 |

---

1. **[5.0 points] Discuss in details 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 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. **[6.0 points] List three different situations lead to the following direct transitions between states:**

**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.

3. **[4.0 points] What are the differences between a short-term and long-term scheduler? (3 points)**

**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. **[5.0 points] 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.

**5. [5.0 points] 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 ISR saves the current CPU context, does some bookkeeping, 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.

**6. [3.0 points] List THREE reasons when parent may terminate the execution of its child.**

**Answer:** The child has exceeded its usage of system resources, the task assigned to the child is no longer required, the parent is exiting and the OS does not allow a child to continue if its parent terminates.)

**7. [3.0 points] Write down TWO of the things that happen when a Unix fork() system call is made at user level. ANSWER**

**Answer:**

1. The operating system creates an identical copy of the process;
2. They share the data and code segments
3. Each has its own heap, stack, and the program counter
4. The child receives a unique process identifier (PID), which is kept in its Process Control Block.
5. fork() returns a 0 to the child and returns the PID of the child to the parent.

8. **[4.0 points] What is the difference between message passign and shared memort as two main models of interprocess communication (IPC)? List TWO examples in which message passing is preferred to be used?**

9. **[5.0 points] Discuss the producer/consumer in your own sentences? Write the required pseducodes?**

**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.