ASSIGNMENT 2

1. (2) RUN to READY can be caused by a priority interrupt

(5) READY to NONRESIDENT occurs if memory is overcommitted, and a

process is temporarily swapped out of memory

(1)READY to RUN occurs only if a process is allowed to run by a processor

(3) RUN to BLOCKED can occur if a process issues an I/O or other

Kernel request.

(4) BLOCKED to READY occurs if the event completes

(perhaps I/O completion)

(6) BLOCKED to NONRESIDENT – It is same as READY to NONRESIDENT.

1. T=22,

P5, P8 – ready/running

P1, P3, P7 – blocked for I/O

T=37,

P1, P3, P8 – ready/running

P5 – blocked suspend

P7 – blocked for I/O

T=47,

P1, P3, P5 – ready/running

P7 – blocked

P8 – exit

1. By fork, a child process will be created with id = 0.
2. Mode switch between threads may be cheaper than mode switch between processes due to the following reasons:
3. It takes less time to create a new thread than a process.
4. It takes less time to terminate a thread than a process.
5. Switching between threads takes less time than switching between processes.
6. Threads enhance efficiency in communication between programs.
7. The advantages of ULTs over KLTs are:
8. Thread switching does not require kernel mode privileges because all of the thread management data structures are within the user address space of a single process. Therefore, the process does not switch to the kernel mode to do thread management. This saves the overhead of two mode switches.
9. Scheduling can be application specific. One application may benefit most from a simple round-robin scheduling algorithm, while another may benefit from a priority based scheduling algorithm. The scheduling algorithm can be tailored to the application without disturbing the underlying OS scheduler.
10. ULTs can run on any OS. No changes are required to the underlying kernel to support ULTs.
11. The disadvantages of ULTs compared to KLTs are:
12. In a typical OS, many system calls are blocking. As a result, when a ULT executes a system call, not only is that thread blocked, but also all of the threads within that process are blocked.
13. In a pure ULT strategy, a multithreaded application cannot take advantage of multiprocessing. A kernel assigns one process to only one processor at a time. Therefore, only a single thread within a process can execute at a time. There is application-level multiprogramming within a single process. While this multiprogramming can result in a significant speedup of the application, there are applications that would benefit from the ability to execute portions of code simultaneously.
14. When a ULT executes a system call, not only is that thread blocked, but also all of the threads within the process are blocked because user process are different from kernel processes. The threads in the user processes are not visible to the kernel. It is said that kernel schedules the process as one unit and gives the unique state to the process. Hence when one thread is blocked whole process is blocked.
15. In multithreaded program, if one KLT makes blocking system call while other KLTs will run. On uniprocessor system, a process that would otherwise have to block for all these calls can continue to run its other threads. If one process is blocked the threads will also be blocked.
16. If a process exits and there are still threads of that process running, they will not continue to run.
17. The difference between competing processes and cooperating processes is
18. There is no interaction between competing processes.

The cooperating processes interact with each other.

1. They are unaware of each other and result of one process is independent of action of other.

They are directly/indirectly aware of each other and the result of one process may depend on the result of other.

1. The difference between strong and weak semaphores is that strong semaphores implement the fairest removal policy (FIFO – process which is blocked longest is released from the queue first) while the weak semaphores do not specify the order in which processes are removed from the queue.
2. Monitor is a software module consisting of one or more procedures, an initialization sequence and local data variables. Local data variables are accessible only by monitor’s procedure and not by external procedures. Process enters monitor by invoking one of its procedures. Only one process can execute in process at a time.
3. The difference between blocking and non-blocking with respect to messages is that for a message
4. If sender is blocked, it cannot send any other message till the message is delivered. If sender is not blocked, it can continue to send messages.
5. If receiver is blocked, it cannot receive any other message till that message is received. If receiver is not blocked, it can continue to receive messages.