OS Assignment-2:

1.

(2) RUN to READY can be caused by a time-quantum expiration or 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 allocated the CPU by the dispatcher

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

kernel request.

(4) BLOCKED to READY occurs if the awaited event completes

(perhaps I/O completion)

(6) BLOCKED to NONRESIDENT - same as READY to NONRESIDENT.

2.At time 22: P1: blocked for I/O P3: blocked for I/O P5: ready/running P7: blocked for I/O P8: ready/running At time 37 P1: ready/running P3: ready/running P5: blocked suspend P7: blocked for I/O P8: ready/running At time 47 P1: ready/running P3: ready/running P5: ready suspend P7: blocked for I/O P8: exit

3.

0

<child pid>

or

<child pid>

0

4. Reasons why a mode switch between threads may be cheaper than mode switch between processes is as follows:

Takes less time to create a new thread than a process.

Less time to terminate a thread than a process.

Switching between threads take less time than switching between processes.

Threads enhance efficiency in communication between programs.

5.The advantages of ULTs over KLTs are:

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.

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.

ULTs can run on any OS. No changes are required to the underlying kernel to support ULTs.

6.The disadvantages of ULTs compared to KLTs are:

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.

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.

7. User process functions separately from Kernel processes. That is, thread structure of a process is not visible to the OS/kernel, which schedules on the basis of processes.

The kernel continues to schedule the process as a unit and assigns a single execution state

(Ready, Running, Blocked, etc.) to that process

Hence once one thread is blocked, the whole process is blocked and consequently all threads in that process are blocked.

8. In a cache system, direct mapping maps each block of main memory into only one possible cache line. Associative mapping permits each main memory block to be loaded into any line of the cache. In set-associative mapping, the cache is divided into a number of sets of cache lines; each main memory block can be mapped into any line in a particular set.

9.No, the threads will not continue to run because termination of a process will terminate all threads within the process.

10. Competing process are unaware of each other that is result of one process is independent of action of other. Whereas cooperative process are directly/indirectly aware of each other and result of one process may depend on the information obtained from others.

11.In strong semaphore the processes which are blocked are released from queue in FIFO(First in first out) order. Whereas in weak semaphore the releasing of processes is not valid.

12.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 a process at a time.

13. Blocking means either a receiver and/or sender is required to wait until the message is received. Whereas non-blocking means either a receiver and/or sender is not required wait until the message is received.

14.Not necessary, busy waiting can be more efficient if the expected wait time is shorter than the time it takes to preempt and re-schedule a thread. This is common on multiprocessors

15.

16.