**Solutions on Threads**

In a scenario in which multiple tasks need to be supported concurrently, we can choose to either create processes or threads for tasks. Which of the following will create lower overhead for the OS? Select the best answer.

C. Incorrect. Creating a unique process for each unique task that runs within its own execution context.
D. Correct. Creating a different thread for each unique task that runs within its own execution context.

Solution: Using unique processes creates more overhead when context switching compared with switching between threads. Furthermore, interprocess communication is more expensive than threads reading and writing to common virtual memory space. Each process requires its own set of resources as well. Thus, it will not result in lower overhead to use processes than threads.

Which of the following is a difference between processes and threads? Select all that apply.

E. Incorrect. Process creation is cheaper and faster than threads.
F. Correct. There can be many threads within a process.
G. Correct. Threads in the same process can communicate directly with each other by writing to the same memory location but processes communication involves the kernel.
H. Correct. Processes has better isolation than threads and one process is less likely to corrupt another process compared to two threads.
Explanation: Process creation is more expensive than threads in general. Hence that option is wrong. The other points are correct. There are many threads within a process, and the best way for them to communicate is by writing to the same memory location. But the shared memory space reduces isolation which is not an issue in processes.

Is the following statement true or false? Select the best answer.

In a web server, creating a thread for each new request from a client and then destroying the thread after the request has been serviced requires less overhead than maintaining a thread pool.

C. Incorrect. True
D. Correct. False

Solution: Creating threads can be expensive so creating a new thread for each request will add a lot of overhead to the web server. It is more efficient to maintain a thread pool and to use a thread from the pool to service each request, returning the thread to the pool after the request has been serviced.

Which of the following are good uses of threads? Select all that apply.

D. Incorrect. Two different applications belonging to two different users, each running in a separate thread.
E. Correct. Web browser has several threads, each supporting a different task (e.g. image rendering, receiving requests from users, sending back HTML responses, etc.}.
F. Correct. An application that breaks up large amounts of data into small chunks for processing in parallel.

Solution: A is not a good way to use threads since processes provide better isolation. It is more desirable to have each user application run in its own separate process. B and C are good uses of threads as they both require running multiple tasks in parallel but yet isolation is not essential in either.

Is the following statement true or false? Select the best answer.

Context switching between user level threads is very fast, requiring only a few instructions in user space.

C. Correct. True
D. Incorrect. False

Context switching between user level threads is indeed very fast, requiring only a few instructions in user space. This is one of the advantages of user level threads.

Which of the following correctly reflects the differences between kernel and user level threads? Select the best answer.

E. Correct. Kernel level threads are all maintained by the kernel, while user level threads are maintained in user space.
F. Incorrect. Kernel level threads that make system calls will block other kernel level threads in the same process.
G. Incorrect. The scheduling of both kernel level threads and user level threads can be customized by the user application.
H. Incorrect. Kernel level thread creation is faster and cheaper than user level thread creation.

It is true that kernel level threads are all maintained by the kernel, while user level threads are maintained in user space. The remaining answers are incorrect. Kernel level threads that make system calls will not block other kernel level threads in the same process. Furthermore, kernel

level threads need to use the OS specific scheduling policy. It is true that the scheduling of user level threads can be customized by the user application. Finally, user level thread creation is cheaper than kernel level thread creation since the kernel need not be involved.

4. Is the following statement true or false?

Threads within a process share the same memory space.
(multiple choice question with ONE correct answer) → 1 point
C. Correct. True
D. Incorrect. False

Solution: Although each thread within a process has its own stack, these stacks are still in the same memory space.