Operating systems

Sheet4

(EED)

Name: Samuel Ayman Shawky

ID: 20010750

Q1)

Discuss the differences between the following:

a. A process and a thread

- **Overhead in creation & deletion.**
- **Way of communication and its speed**

In terms	Process	Thread
Overhead in creation &deletion	Takes along time to create or delete a new process compared to thread time creation	Takes far less time to create or delete a thread in an exist process (about 10X faster)
Way of communication and its speed	use sending messages which is **slow** and inefficient as intervention the kernel is required	Use shared memory and files way which is **fast** and efficient as communication is done without invoking the kernel

Q2) Discuss the differences between the following:

a. User-level and kernel-level threads:

In terms	User-level threads	kernel-level threads
Mapping.	are mapped onto kernel-level threads by user-space libraries,	correspond directly to kernel-level threads.
Dealing with multi-processor systems	are less efficient in utilizing multiple processors,	can be more efficiently scheduled across multiple processors.
Overhead on the kernel	have minimal overhead on the kernel, as thread management is handled in user space.	have higher overhead on the kernel due to kernel involvement in thread management.
Portability	are more portable across different operating systems,	may be less portable due to reliance on kernel-specific APIs.
Who is doing dispatching and scheduling?	dispatching and scheduling are handled by user- libraries	dispatching and scheduling are performed by the kernel.

Q3) Why all threads within a process will be blocked if a user-level thread calls a system call? Does that happen in kernel-level threads?

When thread calls a system call in user level thread all threads of this process will be blocked as it's in user level and share space addresses of process and operating system doesn't know of their existence. It doesn't happen in kernel level thread as it's done through kernel so every thread will continue executing independently of other threads.

Q4) List the advantages and disadvantages of user-level threads over kernel-level threads and vice versa?

	user-level threads	kernel-level threads
Advantages over the another	Light weight and faster switching between each other as it's managed without invoking kernel. ULTs are more portable with other operating systems as they rely on library.	Can block one thread as every thread is executed independently. Can fully utilize multiple processors and achieve perfect utilization.
disadvantages over the another	If UTL called a system call all threads are blocked. ULTs are limited in their ability to fully utilize multiple processors because ULT libraries map ULTs onto a single kernel-level thread.	Slower operation done by KLTs as kernel in invoked. Less portable across different operating systems as it's depending on kernel.

Q5)

What is the faster between the process switch and the thread switch? Explain why.

Thread switch is faster than process switch as threads in same process share memory and resources, so it has fewer overheads, but process switch requires operating system to perform context switching.

Q6)

Would an algorithm that performs several independent calculations concurrently (for example matrix multiplication) be more efficient if it used threads or if it did not use threads? Discuss this point in both user/kernel threads and uni/multiprocessors.

It will be efficient to use threads in several independent calculations.

In ULTs:

There won't be difference between uni\ multiprocessors as library map ULT to a single kernel thread.

In KLTs:

It will be faster if we use threads with multiprocessors, as several threads can run in parallel.

Q7)Explain the states of threads in the windows operating system.

Ready: threads are ready to run and wait to be chosen by dispatcher.

Running: thread is currently executing.

Blocked: thread is waiting for an event to occur.

Finish: when thread complete calculations.

Q8)

A-

Maximum no of jobs at once =20/4=5 jobs

Max no of tapes left idle=16 tapes when one job is in progress

Min no of tapes left idle=0 tapes when 5 job is in progress.

B-

An alternative policy to improve tape drive utilization and avoid system deadlock could be to allow jobs to start with fewer tape (3) drives initially and request additional drives as needed so maximum jobs will be 6 and there will be 2 tapes free to use when it's needed.

Q9)

a. The number of kernel threads allocated to the program is less than the number of processors.

there are insufficient kernel threads to fully utilize the available processors. This can lead to underutilization of CPU resources, as some processors may remain idle while waiting for kernel threads to become available.

b. The number of kernel threads allocated to the program is equal to the number of processors.

Each processor has a dedicated kernel thread, allowing for efficient utilization of CPU resources.

c. The number of kernel threads allocated to the program is greater than the number of processors but less than the number of user-level threads.

With more kernel threads available than processors, the system may experience increased overhead due to context switching between kernel threads.

Q10)

A- What does this program accomplish?

This program creates two concurrent threads. Both threads increment the global variable “my global “in a loop and print characters. The first thread prints 'o' characters, while the secondary thread prints '.'

B- Is this the output you would expect? If not, what has gone wrong?

No, this output is not correct as expected output is my global=40

I think it’s wrong as the two threads start racing to change value of my global and incorrect synchronization in the program.