CS 314 Operating Systems, Spring 2024
Quiz #7 on March 19, 2024

List of the Possible Questions

#1: What are "threads"?

#2: How do threads reduce the high context switching overhead in processes?

#3: How do threads reduce the high overhead of processes when processes try to share data in shared memory?

#4: Show the typical internal structure of a process that consists of multiple threads.

#5: Show the typical internal structure of a thread.

#6: What information do multiple threads in a process can share?

#7: Can threads in a process share global variables in other processes? If yes, briefly describe how. If not, briefly explain why not.

#8: What information must be managed in TCB (thread control block, also known as "private PCB") and what information should be still in the PCB (also known as "global PCB")?

#9: Which of the following items in the PCB for a process should belong to the global PCB or TCB (private PCB)?

(a) Processor registers
(b) Program Counter (PC) register
(c) User ID
(d) Process ID
(e) The list of opened files
(f) The list of the assigned I/O devices
(g) Stack Pointer (SP) register

#10: Which of PCB (global PCB) or TCB (private PCB for each thread) do processor registers belong to? Briefly (but with a proper emphasis on the essential reason) justify your opinion.

#11: Which of PCB (global PCB) or TCB (private PCB for each thread) does Program Counter (PC) register belong to? Briefly (but with a proper emphasis on the essential reason) justify your opinion.

#12: Which of PCB (global PCB) or TCB (private PCB for each thread) does User ID (UID) information belong to? Briefly (but with a proper emphasis on the essential reason) justify your opinion.

#13: Most of the operating systems today includes Stack Pointer (SP) register in their private PCB (i.e., "TCB"). Why is that necessary?

#14: As we discussed in the classroom, "threads" are introduced after many system programmers were using "processes" for multi-tasking (we even discussed that "threads" were introduced to avoid two problems in "processes"). After all, while "processes" and "threads" have many things in common (and "threads" seem to be better than "processes"). Then why do we still use "processes" (mention at least two different reasons)?

#15: What are the two different implementations of "threads"?

#16-01: What are the two modes of processors?

#16-02: What is the primary role of a "kernel mode"?

#16-03: What is the primary role of a "user mode"?

#16-04: Which of the following(s) can a processor in the user mode perform?

(a) Preempt a process (in the running state) in the short-term scheduling.

(b) Read the value of the "real-time clock".

(c) Read the contents in a file at a local storage device

(d) Kill a processes under multitasking OSes

(e) None of the above

#16: What are the advantages in the kernel-mode threads?

#17: What are the advantages in the user-mode threads?

#18: Complete the following table that compares the user-mode and kernel-mode threads.

Factors	User-Mode	Kernel-Mode
Preemptive thread scheduling		
Robustness		
Execution speed		
Portability		