Asa Hayes CSCE-312-199 Ahmed 2 June, 2020

**CSCE-313: Quiz 1**

Part 1: True/False (20pts)

1. The executable image of a program must be loaded into the main memory first before executing. **False**
2. An Operating System (OS) does not trust application programs because they can be either buggy or malicious. **True**
3. There was no concept of OS in first generation computers. **True**
4. The PC register of a CPU points to the next instruction to execute in the main memory. **True**
5. Second generation computers still executed programs in a sequential/batch manner. **True**
6. Time sharing computers gave a fixed time quantum to each program. **True**
7. An OS resides in-between the hardware and application programs. **True**
8. The primary goal of OS is to make application programming convenient. **True**
9. Context switching does not contribute much to the OS overhead. **False**
10. Main Memory access is slower than register/cache access because it is physically outside the CPU. **True**
11. Multiprogramming cannot work without Direct Memory Access (DMA) mechanism. **True**
12. Interrupts are necessary for asynchronous event handling in a CPU. **True**
13. A program can be kicked out of a CPU when it requests I/O operation, or when another Interrupt occurs. **True**
14. A program error can kick a program out of CPU . **False**
15. Interrupts are necessary to bring a program back to CPU if it was previously kicked out. **True**
16. The “Illusionist” role of the CPU allows a programmer write programs that are agnostic of other programs running in the system. **False**
17. Modern operating systems come with many utility services that are analogous to the “Glue” role of the OS. **True**
18. Networking service is not a core OS part, rather a common service included with most OS. **True**
19. Resource allocation and Isolation are not part of the core OS, rather common services included with OS. **False**
20. Efficiency is the secondary goal of an OS. **True**

Part 2: Short Answer (80pts)

1. [10 pts] Define multiprogramming. How is this better than sequential program execution?

Multiprogramming, much as it sounds like, is the process of running multiple programs concurrently. With sufficient memory access, the CPU can run a program/programs while the previous program is being worked on elsewhere. Otherwise, if restricted to sequential programming, the CPU would not be able to multitask at all, leaving the CPU idle while processing a program elsewhere.

1. [10 pts] Define time-sharing. Can you combine time-sharing with multiprogramming?

Time sharing allows for the appearance of having multiple programs run at the same time in the same place via switching back and forth between interruptible programs at a set (very small) time interval. Using a timeshare structure on a multiprogrammed system makes it to where the ‘pieces’ of the running programs are made to fill the spaces between each other by making having the switching be scheduled to where no piece overstays its welcome and can be sized to fit between more regularly scheduled pieces.

1. [10 pts] Say you are running a program along with many other programs in a modern computer. For some reason, your program runs into a deadlock and never comes out of that. How does the OS deal with such deadlock? How about infinite loops? How does the OS detect, if at all, such cases?

The OS handles deadlocks via exceptions. If a program hits a deadlock to where it keeps hitting a problem that prevents it from continuing in any circumstance, an exception will be thrown that either adjusts things to fix the issue or aborts the program to stop it from continuing to take space from properly working programs. There are scenarios to where an error may not throw an exception, such as an infinite loop, that would not throw an exception. For infinite loops however, the system’s timer stops loops from continuing for too long and monopolizing resources. But, even though it may stop the infinite loop from crashing the system, an infinite loop is still technically running properly even though it won’t go anywhere, so the system wouldn’t know there was a problem and just continue switching back to it when its allocated time comes back up on the timer. This is usually resolved by the loop eventually creating a memory error, but not always.

1. [25 pts] In a single CPU single core system, schedule the following jobs to take the full advantage of multiprogramming. The following table shows how the jobs would look like if they ran in isolation. [Use the attached pages from W. Stallings book to solve this problem]

|  |  |  |  |
| --- | --- | --- | --- |
|  | JOB1 | JOB2 | JOB3 |
| Type of job | Full CPU | Only I/O | Only I/O |
| Duration | 5 min | 15 min | 10 min |
| Memory required | 50MB | 100MB | 75MB |
| Needs disk? | No | No | Yes |
| Needs terminal? | No | Yes | No |

a. What is the total time of completion for all jobs in sequential and multi-programmed model? **Sequential: 30 minutes. Multiprogrammed: 15 minutes.**

b. Fill out the multiprogramming column in the following table (i.e., when the jobs are scheduled in multiprogramming). Assume that the system’s physical memory is 256MB.

|  |  |  |
| --- | --- | --- |
| Average Resource Use | Sequential | Multiprogramming |
| Processor | 5/30 = 16.67% | 5/15 = 33% |
| Memory | 32.55% | 61.85% |
| Disk | 33.33% | 66.67% |
| Terminal | 50% | 100% |

Memory usage is computed as follows: (5minx50MB + 15minx100MB + 10minx75MB) / (30minx256MB) = 32.55%

Other resources are fully utilized during the time they are utilized. So, you compute utilization only based on the duration they are used.

5. [25 pts] Consider the following program and provide explanations where asked in the code comment after running the program in your system. Note that there are 8 places where explanation is needed. For explanation 1 (i.e., the first commented line), does the Header size equal the sum of individual data types (i.e., a char, an int and a pointer)? Try to explain this with something called “packing”.

[File with filled code and comments is included alongside this one as HeaderSizes.cpp.]