CO502 Operating Systems – Tutorial 1 Solution
Introduction & Processors

Morris Sloman

Note that the solutions below only briefly list (some of) the key points that should be included in an answer. They are by no means complete. In an exam, you are expected to spell out the solution more fully and include a detailed explanation of your reasoning.

1. The issue of *resource allocation* shows up in different forms in different types of operating systems. List the most important resources that must be managed by an operating system in the following settings:

   (a) Supercomputer

   Solution: Processors, memory — any resource that affects the computational performance of the supercomputer.

   (b) Workstations connected to servers via a network

   Solution: The network is a resource that is shared by all connected computers. Access to it may be managed by the OS through the network protocol stack that avoids congestion using protocols such as TCP.

   (c) Smartphone

   Solution: A smartphone may have limited battery life, so battery power is a resource that has to conserved by the OS. For example, the OS may decide to put the CPU to sleep or switch of certain unused hardware features. The cellular network is a potentially costly resource so access to this may also have to be controlled and limited to authorized applications.

2. What is the kernel of an operating system?

   Solution: The part of the operating systems that is always in memory and implements the most commonly executed functions of the OS. The OS kernel executes in kernel or privileged mode and therefore has complete access to all hardware (in contrast to user-mode processes).

3. Why is the separation into a user mode and a kernel mode considered good operating system design?

   Solution: A process executing in user-mode can cause less "damage" because it is restricted from executing privileged operations. By splitting an OS design into user-mode and kernel-mode components, the OS designer makes it explicit which parts of the OS require raised privilege and full access to the hardware. This is an example of the "principle of least privilege" that should guide the design of any secure systems.

   Give an example in which the execution of a user processes switches from user mode to kernel mode, and then back to user mode again.

   Solution: A system call (using a trap instruction) is an example of this.

4. A portable operating system is one that can be ported from one system architecture to another with little modification. Explain why it is infeasible to build an operating system that is portable without any modification.

   Solution: By definition an OS has to interact with the hardware directly. It therefore contains code that depends on the specifics of the processor architecture such as its instruction set and the memory management system.

Describe two general parts that you can find in an operating system that has been designed to be highly portable.

Solution: A portable operating system can be split into a (1) platform specific and into a (2) platform independent part.

The platform-specific part contains any OS code that is dependent on the given processing architecture and platform. All other OS functionality is implemented as part of the platform independent part that uses the API exposed by the platform-specific part to invoke any low-level hardware functionality. When porting the OS, only the platform-specific part has to be reimplemented for a new platform. The rest can just be recompiled for the new architecture.

This type of division can be found in the Linux kernel, which makes it (relatively) easy to port it to new platforms.

5. Why do most Operating Systems provide multiprogramming?

Solution: I/O is slow compared to computation time and so programs spend a lot of time waiting for I/O. Multiprogramming allows the processor to be used by other users while one program is performing I/O. It improves the utilisation of expensive resources.

Multiprogramming also allows the user to perform different activities at the same time eg. compilation of one program while editing another or printing while doing something else. Needed to allow one process to create other "child" processes as in Unix.

6. Explain why multiprogramming systems require:

a) Hardware interrupts from I/O devices
b) Independent Direct memory access channel

Solution: Without interrupts the OS would have to regularly poll I/O devices to determine whether the I/O is complete. If Program A was waiting for I/O and Program B had the processor, Program B would have to include regular calls to the OS to tell it to check if any other programs I/O was complete. This is messy and not very reliable.

Without DMA channels the OS would be so busy performing I/O to fast devices like discs, that it would not be feasible to allocate the processor to another program.

7. If a multithreaded process forks, a problem occurs if the child gets copies of all the parent's threads. Suppose that one of the original threads was waiting for keyboard input. Now two threads are waiting for keyboard input, one in each process. Does this problem ever occur in single-threaded processes?

Solution: No. If a single-threaded process is blocked on the keyboard, it cannot fork.

8. What is the biggest advantage of implementing threads in user space?
What is the biggest disadvantage?

Solution: The biggest advantage is efficiency. No traps to the kernel are needed to switch threads. The ability of having their own scheduler can also be an important advantage for certain applications. The biggest disadvantage is that if one thread blocks, the entire process blocks.

9. If in a multithreaded web server the only way to read from a file is the normal blocking read() system call, do you think user-level threads or kernel-level threads are being used? Why?

Solution: A worker thread will block when it has to read a Web page from the disk. If user-level threads are being used, this action will block the entire process, destroying the value of multithreading. Thus it is essential that kernel threads are used to permit some threads to block without affecting the others.

10. Why would a thread ever voluntarily give up the CPU by calling thread yield()?
After all, since there is no periodic clock interrupts, it may never get the CPU back.

    Solution: Threads in a process cooperate. They are not hostile to one another. If yielding is needed for the good of the application, then a thread will yield. Usually the same programmer writes the code for all of them, so knows when yield is required

11. The register set is a per-thread rather than a per-process item. Why? After all, the machine has only one set of registers.

    Solution: When a thread is stopped, it has values in the registers. They must be saved, just as when the process is stopped the registers must be saved. Multiprogramming threads is no different than multiprogramming processes, so each thread needs its own register save area.

12. In a system with threads, is there one stack per thread or one stack per process when user-level threads are used? What about when kernel threads are used? Explain.

    Solution: Each thread calls procedures on its own, so it must have its own stack for the local variables, return addresses, and so on. This is equally true for user-level threads as for kernel-level threads.

13. Would an algorithm that performs several independent CPU-intensive calculations concurrently (e.g., matrix multiplication) be more efficient if it used threads, or if it did not use threads? Why is this a hard question to answer?

    Solution: The answer to this question depends on the implementation of threads and if the system is a uniprocessor system or a multiprocessor system. If the system is a multiprocessor system, the implementation using threads is more efficient if each thread could execute on a separate processor. However, if the system is a uniprocessor system, the implementation using threads would be less efficient due to the additional overhead incurred by spawning threads and switching contexts between the threads.

Mentimeter Questions

1) Which is not the function of the Operating System?

    d Virus Protection

2) Non-determinism results from:?

    All the above

3) Which of the following is not true?

    c) access to disk interface is a privileged instruction – only accessible to Kernel

Multiprogramming and Processes

1) The objective of multi-programming is to : (choose two)
    a, d Have some process running at all times & To maximize CPU utilization

2) How many processes in memory waste 10% of CPU if 80% process time is waiting for IO
    CPU utilization = 90% = 0.9 = $1 - p^n$ where p = 0.8
    $1 - 0.8^n = 0.9$ $0.8^n = 0.1$ $n = \log_{0.8}0.1 \approx 10$

3) What does initial process print (ignore new lines)
    XYZ

4) Stacks: Which of the following is correct?
    b, d Both kernel and user threads have own stacks

5) Algorithm performs concurrent CPU-intensive calculations
    see 13. above