Assignment 2
Jacob Hillebrand
Operating Systems
Part 1: Written Questions

1. What is the difference between Multiprogramming and Timesharing Operating systems? What are the problems (challenges) in each of them?

   - Multiprogramming is the idea of simply queuing up jobs or programs for the processor so that, when one program reaches a point where it needs to wait for I/O, the processor can switch to the next job instead of waiting for the I/O on the first program so as not to have an idle processor. This is good for processor utilization, but could cause program response delays if the CPU is in the middle of another program that has not hit an I/O wait yet. Timesharing, on the other hand, is the idea of dividing up processor time between users, not programs. Each user has processor time they can interact with system in, and the OS interleaves the execution of each user's program in short bursts. This means that the system is very responsive to user and program requests, but may be under-utilizing the CPU in the event that one user is idle during their "timeslot". (Source: Lecture 3 slides)

2. What is the Operating System Kernel?

   - The Operating System Kernel Controls execution of the processors. Essentially, this means the kernel organizes things like thread scheduling, process switching, exception and interrupt handling, as well as multiprocessor synchronization. Because the Kernel manages these things, its own code cannot run in threads. (Source: Textbook *Operating Systems Internals and Design Principles*)

3. One of the operating system roles is acting as an illusionist. Explain what that means and how that works using some examples?

   - The idea that an operating system acts as an illusionist refers to the fact that the Operating System must provide high-level abstractions from the computer's hardware devices to the programs running on the computer. For example, not every program may be aware of exactly what the procedure is to write to and read from a hard drive, and cannot directly instruct the CPU to do so. However, the operating system is aware of this procedure, and simply provides "write" and "read" system calls that the running programs can easily utilize to perform their desired actions without a deeper understanding of how the computer's hardware works. (Source: Lecture 3 slides)

4. What are the drawbacks of the Programmed I/O and Interrupt based I/O? Why the second one is more efficient?

   - Both Programmed I/O and Interrupt-based I/O have drawbacks in the form of additional CPU cycles. For example, Programmed I/O will waste CPU cycles waiting on I/O devices. I/O devices are significantly slower than CPUs, so if a CPU sends an instruction to an I/O device and needs to wait on a response, the CPU will be wasting resources waiting for the slower I/O device. Interrupt I/O will also result in additional CPU cycles, as it will need to utilize CPU cycles processing interrupt requests and then branching to fullfill them. However, the cycles required for processing interrupts are generally significantly fewer than the cycles required to wait for an I/O device to respond, so Interrupt-based I/O is much more efficient. (Source: Textbook *Operating Systems Internals and Design Principles*)

5. Which of the following are likely components of an operating system?

   - b) File System
   - e) Device Driver
   - f) CPU Scheduler

6. What is Dual-Mode operation in an operating system? How would the Dual-Mode technique support system Security and reliability? Is hardware support needed to implement the Dual-Mode technique?

   - Dual-Mode operation refers to the idea that an operating system has two modes of operation; User Mode and Kernel mode. In user mode, user programs can be executed, but there are certain areas of memory that are inaccessible and certain instructions that cannot be executed. In kernel mode, privileged instructions can be run and the aforementioned protected areas of memory can be accessed, but user programs cannot be directly run. This approach is taken to support system security and reliability. For example, because kernel mode protects critical memory and instructions, the system cannot be (accidentally or intentionally) compromised by a bad user program. This also improves system reliability, as it allows the user to execute programs, but in a safe manner. However, hardware support is needed to implement this system, as there must be a "Mode bit" to provide the ability to distinguish whether the system should be in User mode or kernel mode. (Source: Lecture 3 Slides)

7. What are the main three reasons for the kernel to take control from a user process?

   - The kernel will take control from a user process in any of the following three events:
     - Interrupts - triggered by timer or I/O devices
     - Exceptions - unhandlable events caused by unexpected (or malicious) program behavior
     - System Calls - request from user program to execute protected procedure calls

     (Source: Lecture 3 Slides)

8. How does the processor know what code to run when an interrupt occurs?

   - There are a special set of instructions that each CPU automatically executes in the event of an interrupt. These instruction sets do two things:
     (a) Determine what kind of interrupt occurred
     (b) Execute a specific instruction set to properly handle the interrupt and return the system to normal operation

     (Source: Textbook *Operating Systems Internals and Design Principles*)

9. Which of the following instructions should be privileged?

   (a) Set value of the timer - PRIVILEGED
   (b) Read the clock - non-privileged
   (c) Clear memory - PRIVILEGED
   (d) Issue a trap instruction - non-privileged
   (e) Turn off interrupts - PRIVILEGED
   (f) Modify entries in the device-status table - PRIVILEGED
   (g) Switch from user to kernel mode - non-privileged
   (h) Access I/O device - PRIVILEGED

   (Source: geeksforgeeks.org)

10. Give two reasons why caches are useful. What problems do they solve? What typical issues are associated with it?

    - Caches can be useful for two reasons. First, caches increase the speed of memory access because disk drives, and even main memory, are significantly slower to access than cache memory. Thus, if commonly used memory is loaded into the caches, it means fewer wasted CPU cycles in waiting for the data to be loaded into the processor from main memory. Secondly, caches can be leveled, and support the principle of locality. This means that commonly used pieces of memory can be stored in the level of cache or memory corresponding to the amount of times the CPU needs to access it. In this way, the miss rate can

be reduced, which reduces the miss penalty and the amount of wasted CPU cycles on a miss. However, there are a few issues typically associated with caches. Caches can be very expensive to manufacture, and are typically very small. As such, additional CPU resources are required to load data to and from the various tiers of caches to make it accessible. (Source: Class notes)

11. What is DMA (Direct Memory Access)? How does it affect CPU performance?

    - Direct Memory Access (DMA) is the idea of using a dedicated memory controller to load data between the disk drive and main memory of a system. This means that the CPU doesn't need to use cycles to perform this task (it frees up the resources for other tasks) which can increase CPU performance. (Source: Class notes)

12. Since you learned in class what system calls means, answer the following question: **On a 64-bit Linux-based OS, which system call is used to:**

    - Send a signal to a process - s = kill(pid, signal)
    - Set the group identity of a process - s = setpgid(pid_t pid, pid_t pgid)
    - Mount a file system - s = mount(special, name, flag)
    - Read/Write system parameters - s = _sysctl(struct __sysctl_args *args)

    (Sources: Lecture 3 Slides, setpgid man page, sysctl man page)