YOUSER ALALUSI

HomeWork 1

Due Date: Wednesday 9/22/2021
HW Delivery: submit on **Canvas** by the due date, before midnight
Total Points: 60

General Rules: Create homework, compose specifications or any text by using a common *document-creation* tool, such as Microsoft® Word. Each HW question is worth 3 points. To complete your work, consult lecture notes and the www.

1. What is essential about an Operating System?
    - A computer's operating systems is a highly complex piece of system SW. Acting as intermediary between computer user and HW plus SW services. It is essential because it decides between conflicting requests for efficient and fair resource use. OSS is a control program that controls execution of programs to prevent errors and improper use of computer.

2. Why is it meaningful in a computing environment to use an Operating System (OS)? What key added value does an OS provide?
    - The current trend is toward providing more ways to access these computing environments. Web technologies and increasing WAN bandwidth are stretching the boundaries of traditional computing. Companies establish portals, which provide web accessibility to their internal servers. Network computers which are essentially terminals that understand web based computing – are used in place of traditional workstations where more security or easier maintenance is desired. Networking becoming ubiquitous even home systems use firewalls to protect the electronic brain from internet attacks.

3. While technology evolved, some early computer products did not include an OS. What is a possible constraint, value and use of a computer system without OS?
    - In today's world the computer without an operating system is to difficult and the execution of the program is itself a challenging task. Without OS, the following are the limitations: each program requires own driver for video card, sound card, and hard drive. On the view of programmers, it is too hard to program at the machine level or assembly level. On olden days, if the user needs to use the computer the first step would be to sign up for the time on the machine. If the entered time is reached, then again the user needs to feed in a deck of punched card. Additionally, the user has to load the compiler manually before loading the actual program. The first step is to load the compiler before loading the program into computer. Nest, the bootstrap loader includes the initial set of cards in the input deck and this causes the rest of the cards to load into the system. Now, the user compilers the program, if there is error in the code then the user needs to repunch the card from the beginning and feed the deck into computer again to compile the program in another attempt. If the program is compiled without any errors, then link the code into object code with library files to create the executable file. These are the steps to load and execute the program without using an OS.

4. Explain the essence of a true **interrupt**. What does cause interrupts? How can a user application "know", that is has been interrupted?

    - An interrupt interrupts regular execution: transfers control to the computer's interrupt service routine, through so called interrupt vector, which contains the address of each distinct service routine. Interrupt architecture must save the address of the interrupt instruction and the complete state of the interrupted SW. a trap or execution is a software generated interrupt caused by an error or a user or SW request. At interrupt, OS preserves the CPU state by storing registers, program counter, stack, etc. OS determines, which type of interrupt can occur, polling, or vector interrupt system.

5. List some key "management" functions (at least 5) of an OS?
    - Process management
    - Memory management
    - File system management
    - Free-space management
    - Storage management
    - Device management

6. Characterize the type of information residing in main memory, and contrast this with information residing on secondary storage.
    - Main memory: large, addressable medium to store information in the CPU accesses via loads and stores. Random access memory, hence, name: RAM. Typically volatile, bus fast compared to peripheral device.
    - Secondary storage: extension of main memory that provides large nonvolatile storage capacity.

7. What is the meaning of the acronym EULA? Why use EULAs? Discuss the typical programmer's cost of EULAs for Unix and Linux.
    - EULA is an end-user license agreement that generally protects SW companies from errors in programs. Alongside acting as a platform for granting licenses, a EULA is also super valuable in protecting your right to revoke licenses. This is because the EULA can include a clause stating that you, as the vendor, have the right to susped or terminate licenses and associated rights. Unix is originally a command-line based, textual input and output based OS meanwhile, Linux a later Unix "cousin", is distributed and supported free, free of license fees.

8. Characterize an Operating System's function of: Resource allocation, resource sharing, accounting, protection, and security
    - Resource allocation is when multiple users or multiple jobs run concurrently, resources still are individually allocated. Accounting is to trach which users consume how much and which kind of compute resources; user can be piece of SW. protection and security is the owners of information in multiuser or networked computer system wish to control use of that information; concurrent processes should not interfere. Lastly, resource sharing is when threads share resources (globals, files, etc.) of process, easier than shared memory or message passing alternatives.

9. Describe ***System Call***. How can system calls be made to fit a user's specific needs? List a sample system call.
    - System calls changes mode to kernel, return from system call resets it to user mode. System calls, which are allowed requests for help from OS, demanded by user program. An example of a system call is to copy contents of one file to another file. System call interface invokes intended function in OS kernel; returns status and function return value. Transparent to caller/user how system call is implemented.

10. Explain in which programming language an OS could be implemented. Argue Why?
    - Early OSes implemented in assembly language then in higher level languages like Jovial, PL/1, and now commonly C++. Emulation allows OS to run in non-native environment, typically slow execution, and ease of porting to another host.

11. Outline the **hierarchy** of at least 5 different **storage** resources and technologies for holding information; mention (or sort by) relative speeds and capacities.
    - The five hierarchies in the memory are registers, cache, main memory, magnetic discs, and magnetic tapes. The first three hierarchies are volatile memories which means when there is no power and then automatically, they lose their stored data. Whereas the last two hierarchies are not volatile which means they store the data permanently. The ability of the memory hierarchy is the total amount of the data the memory can store, because whenever we shift from top to bottom inside the memory hierarchy, then the capacity will increase. When we shift from bottom to top inside the memory hierarchy, then the cost for each bit will increase which means an internal memory is expensive compared to external memory.

12. Give an informal definition for OS “kernel”.
    - The kernel is a computer program at the core of a computer’s operating system and has complete control over everything in the system. It is the portion of the operating system code that is always resident in the memory and the facilities interactions between hardware and software components.
    - “The once piece of OS SW running at all times on the computer” is the kernel.
    - “Portion of the operating system code that is always resident” in memory. Generally, kernel is first programs loaded, right after booting, through special device.

13. Describe in detail what is essential in a **Real Time** compute environment.
    - Real time often used in embedded systems. Vary considerable, special purpose, limited purpose OS, real time OS. Many other special computer environments as well such that some have OSes, some perform tasks without an OS. Real time OS has well defined fixed time constraints. Processing must be done withing that time constraint, correct operation only if constraints are met, and if time constraints are not met, generally catastrophic consequences such as data loss.

14. What are essential steps to get an Operating System to work at the moment a computer is turned on (AKA **booted** up)?
    - When powering on aka booting the system, OS execution starts at a fixed memory location. First, firmware ROM holds initial boot code, to load OS part. Similar to C or C++ program, starting at point of main(), the OS begins at _start: code in crt0.o provides that label _start: so other OSes of course use different, but similar convention. To get OS started, execute boot procedure: small piece of code-bootstrap loader, stored in ROM or EEPROM locates the kernel, loads it into memory, and starts. Sometimes two step process: boot block at fixed location is loaded by ROM code; then loads bootstrap loader from disk.

15. Characterize in detail what is a “process”. Describe: typical process needs; types of processes; cause of process start and process end.
    - Process is a program in execution which forms basis for computations, managed by an OS. It’s a program that is executing, code that is currently running or ready to run but in the wait queue. Needs and features of processes such as scheduling, priorities, creation, execution, interruption, termination, and communication. While process executes, overtime it changes state. New is when process is created by loading from disk into memory, running is when instructions are being executed on a CPU, waiting is when process is waiting for some event to occur, could be data input, or another event; may change priority while waiting, ready is when process is waiting to be assigned to a processor as it has not been granted a CPU, and terminated is when process has completed execution its while memory space is being freed and all other resources will be reacquired by OS.

16. What is an "idle Loop"? Why is it part of an OS? Why should there be an Idle Loop?
    - Executing the idle loop is the part of the OS code which recognizes that there are no user request needed to be handled right now. If the CPU supports idle states allowing it to draw less power while not executing any instructions, the idle loop invokes a CPU idle governor to select the most suitable idle state for the CPU and it puts the CPU into the selected idle state with the help of a CPU idle driver. Generally, the idle state selection carried out by the CPU idle governor is based on predicting the duration of the idle time for the CPU, so it is not deterministic.

17. What is a "utility program?" Why have one? Give an example. Must the utility program be an integral part of the OS?
    - Utility program is a small program that performs many of the general housekeeping tasks for the computer such as system maintenance and file compression. Serve as tools for doing system (SW, HW, FW) maintenance and repairs not handled by the OS. Utilities make it easier for users to create, copy, delete, store files on storage devices. It also repairs (some, sometimes) damaged data files, translates files so different programs can process them, guard against viruses and other harmful programs, guard against viruses and other harmful programs, compress files so they take up less space, and communicate between different computers, locations. Examples of utility programs are antivirus software, backup software and disk tools. Antivirus software helps to protect a computer system from viruses and other harmful programs. A computer virus is a computer program that can cause damage to a computer's software, hardware or data.

18. Explain "disk fragmentation?" What causes it? What is the impact on user programs?
    - Disk fragmentation occurs when a file is broken up into pieces to fit on the disk. Because files are constantly being written, deleted, and resized, fragmentation is a natural occurrence. When a file is spread out over several locations, it takes longer to read and write but the effects of fragmentation are far more widespread.

19. What is DMA? Why does a computer system support DMA? What is its main advantage?
    - Direct memory access is IO managed by OS, minimal CPU support. Used for high-speed IO devices able to transmit information fast. Device controller transfers blocks of data from, buffer storage directly to or from main memory without CPU intervention.

20. List some similarities and differences between the Unix and Linux operating systems?
    <u>Similarities:</u>
    - Both are capable of true multitasking.
    - Both run on a variety of hardware platforms.
    - Both are highly modular.
    - Both are capable of using both a command line and a GUI.
    - Both support multiple users and support different privileges between users.

    <u>Differences</u>:
    Unix
    - A family of multitasking, multiuser computer operating systems that derived from the original AT&T Unix.
    - Developed by a group of employees including Ken Thompson, Dennis Ritchie and Brain Kernighan.
    - Source code is not available to the general public.
    - Contains the command line interface.
    - Used for servers, workstations, mainframes and high-end computers.
    - Not portable.
    - Installation requires more sophisticated high-end hardware.

- Expensive.

Linux

- A family of free and open-source software operating systems built around the Linux kernel.
- Developed by Linus Torvalds.
- Source code is available to the public.
- Contains the command line and graphical user interface.
- Used for personal computers, desktops, and used for game development, embedded systems, etc.
- Portable and can be executed on various hard drives.
- Does not require more specific hardware components.
- Free.