Lucas Sarweh
sarwehl@uwindsor.ca
ID: 110042658

Midterm Exam

Question 1:

Opcode - Opcode is the language that is unreadable to humans, but is what the machine understands. It is binary numbers where each line executes a task and the opcodes are executable line by line. It list the operation and then the parameters.

Operating System - The operating system is what lets a computer scheduale and manage multiple processes for a user. It takes the machine away from the user so they can focus on solving real world problems easier. The operating system is programmed at a system-level.

Kernel - The kernel manages the computer and is part of the operating system. It is the mediator between the CPU, Memory, and devices and the tasks that must be executed.

Built-Ins - The built-in commands that are a part of the shell. These commands can answer questions or perform a task.

Instruction Pointer - The instruction pointer is what tells the CPU what memory address to process next. It can be modified by things like the OS.

MBR - The master boot record holds where the operating system is located and tells the IP. It is located at the very beginning of the boot drive.

Program vs Process vs Processor - A program is a set of instruction for the computer that are written and stored in long term memory. A process is a program that has been loaded into memory for execution. The processor is the hardware in the computer that takes the steps to execute a process.

System Call - A system call is when a process needs to perform a task that it is not able to do itself, so it asks the kernel to do it for it.

Platform-free Program - A program that is free to use and to modify. One can modify it and rerelease it as a seperate program.

File Descriptor - This is the dynamic connection between a process and a file it is using. The file descriptor changes each time a new one is needed and is the case even if you open the same file twice. The OS assigns it.

Question 2:

Explain the trip that a mouse move takes to an application-level program.

First a displacement is made using the pointer on the mouse and is sent to the processor. Then, the Interrupt Request is made by the processor when it recieves the signal. A lookup table will be referenced because each interrupt has a different shock represented by a number. In this case it will use the mouse movement and the instruction pointer will receive the address to the interrupt request handler for mouse move in the kernel.
Then the handler will determine what to do with the input and return a result to the application.

Question 3:

Zahra has only one hard disk drive (HDD) but wants to switch between UNIX and macOS at the system bootup. As a programmer, Zahra should write what program and at what level of programming?

Zahra needs to write at a system-level because in order to be able to have the choice to switch between UNIX and macOS you have to modify the MBR. The machine code in the MBR gives the address of the OS. So, you would have to program a way to change the address given on startup such as holding down a key on the keyboard to switch to your secondary operating system.

Question 4:

POSIX provides the specification for a standard operating system based on the C programming language. Does this mean that an operating system must be implemented in C in order to be POSIX-compliant? Justify your answer.

I think that as long as when you try to do a POSIX standard operation and it does it, it is complying with POSIX. It is defined in the C programming language but every language will boil down into opcodes. So you can be POSIX compliant without using C, but it will be harder to find the equivalent libraries and also it may be a slower operating system because C is the fastest high-level language.

Question 5:

After a successful run of a process, how and why the processor gets back to the shell? Who does that?

The process that is running is the one that must give back control to the shell because it has control of the instruction pointer. So therefore the process does it. The way it does it is before the process ran the kernel attaches prologue and epilogue code to the beginning and end of the process. This code contains the adress to get back to the shell.

Question 6:

Let’s assume sh shell accepts scripts. Is it able to run bash scripts on sh shell? How about the reverse, is bash shell able to run sh scripts? Justify your answer.

For all POSIX compliant commands that are used the sh and the bash should be able to execute them. However, the bash shell although POSIX compliant, has added functionalities that the base sh shell does not. So, the sh shell cannot run bash scripts. But, the bash shell will be able to run sh scripts because the sh only has POSIX commands which the bash shell also has.

Question 7:

Yao wants to log the time elapsed of any programs, that is Δtime = finish time – start time, inside a computer system. Which part of the operating system below should be changed to support Yao’s need, and how? a) Shell b) Library Routines

The library routines of the operating system need to be modified. A routine that starts everytime a program is about to be run that will calculate delta time by: First, getting start time when the program first starts, Second, when the process ends get the end time from the system, Finally, calculate delta time = finish time - start time and then log it inside a different file.

Question 8:

Can you think of an identifier for a file instead of file descriptor? Justify your idea.

Instead each file is given a unique number based off of the size of the permanent

storage device installed.
In order to access the file a random math operation/function is performed using that unique number and the program
must know what that math operation/function is in order to use the files ID to get the result. The result
would be the path to read/write the file. The math function is given upon opening a file.