Operating System Exam 1

29th November 2021

Tanmay Garg CS20BTECH11063

Q1. System calls are executed in the kernel-mode (supervisor or privileged mode) by the Operating System to satisfy some request on behalf of the normal user. In this case, how does the OS ensure safety of the system when it is executing a system call in the privileged mode on the behalf of the user. As you might recall normally all the user programs run in user-mode for greater security?

Sol.

- A safety Protection system is designed by making some of the machine instructions as privileged instructions
- These instructions may cause harm to the system if run directly
- The hardware is designed to allow privileged instructions to be performed only in kernel mode
- If the program attempts to execute a privileged instruction in user mode, then the hardware won't execute the instruction
- This process would then be flagged as illegal and traps inside the OS
- Programs communicate by using system calls with the OS
- The OS creates various interrupt service routines (ISR) based on the trap mode requested by the user program
- These ISR are independent of the user programs and are handled by the OS

Q2. For each of the following system calls, give a condition that causes it to fail: fork, exec.

Sol.

The conditions where fork () and exec () can fail:

fork ():

- The fork process can fail when no child process could be created
- This may be due to parent process might be accessing a reserved memory or performing an illegal expression on some variable
- Another situation may be when parent process tries to create a greater number of processes than the hardware could allow based on the number of parallel processes enabled for any user program
- Another situation may be when user program tries to access more memory than available

exec ():

- The exec process can fail when the child process tries to access a path which is not present
- Another situation may be when the child tries to access a path which has been denied access to by OS for any user program, such as specific system files
- Another situation may be when the child process tries to execute a binary or an operation which in itself is illegal

Q3. Virtual machines have become very popular for a variety of reasons. Nevertheless, they have some downsides. Name atleast one such disadvantage.

Sol.

The disadvantages of using a Virtual Machines are:

- The downside of virtual machines is that the resources must divided between Host OS (the main OS of the system) and Guest OS (the secondary OS)
- Virtual Machines cannot take full advantage of a computer's hardware as the instructions needed to be given have to pass through a series of optimizations and conversions by the Virtual Machine Manager
- The Performance of the VM will be much lower than running the exact same configuration provided to the VM OS and running that OS on bare metal hardware
- There may also be a case when there might not be enough hardware present for the VM configuration such as a smaller number of CPU cores or a smaller number of threads are present in actual hardware whereas VM demands for more
- This may cause either of the OSes to crash when a heavy-duty task is being performed on either Guest OS or the Host OS leaving lesser hardware and resources available to either of the OSes
- Privileged instructions generated in VM OS take additional time to run

Q4. Using the program shown in Figure below, explain what the output will be at lines X and Y. Please justify your answer.

Sol.

- Nums was initialized to {0, 1, 2, 3, 4} at the beginning of the program
- When main() function starts in the loop it gets initialized to {1, 1, 1, 1, 1} in the parent process
- fork() procedure is called:
    - Child process begins executing its own process. It has a copy of the array num which holds the value {1, 1, 1, 1, 1}
    - Child process performs its for loop
    - Array num is now updated to the values {1, 0, -1, -2, -3}
- Parent Process waits for the child to be completed
    - Parent process has a copy of the array num which hold the value {1, 1, 1, 1, 1}
    - Parent process performs its for loop
    - Array num is now updated to the values {0, -1, -2, -3, -4}
- The Output at line X
    - CHILD: 0 1CHILD: 1 0CHILD: 2 -1CHILD: 3 -2CHILD: 4 -3
- The Output at line Y
    - PARENT: 0 0PARENT: 1 -1PARENT: 2 -2PARENT: 3 -3PARENT: 4 -4

Q5. Figure 3.5 (page 113 of the book) suggests that a process can only be in one event queue at a time. Is it possible that you would want to allow a process to wait on more than one event at the same time? Provide an example.

Sol.

- Yes, it is possible to make a process wait on more than one event at the same time
- This situation can occur in the case of servers where single server computer may be dependent on the other server computers for some data
- In that case, multiple processes have been created and the main computer then waits for all those processes to finish before it can move ahead and complete its own process
- Another example can be when, a machine learning model needs to be trained
- In that case, multiple mathematical operations are being performed directly on the CPU and the GPU and all these processes are independent of each other or may dependent also
- The main program requests for a matrix with some values and also needs to load some values into the matrix, so, it waits for both the GPU and CPU to give the main program the matrix it needs and the values it needs to store in them or any sort of operations that needs to be performed upon them
- After receiving both the things, it then again sends it either to the GPU or the CPU to perform operations based on how the model needs to be trained
- In the meanwhile, the main program can also request for another model or another mathematical operation that needs to be performed on some other variable while the first task is being performed but, in this case, both the processes are independent and don't depend on each other so they can be performed parallelly till both of them give some output and then it can be loaded into the main program for further processes that need to be done
- So, when the main process is executed, somewhere during its execution it will need to wait for outputs from both the CPU and GPU operations to move ahead
- In the case of a normal program, which doesn't involve heavy mathematical operation, can be achieved by having structs or classes of the child programs
- This class may store the functions of all the child processes that need to be performed and store their PIDs as well