Problem 1: OS Overview and Boot Process 1.1 (3 pts) What are the THREE roles of an operating system? For each role, give an example as well.

Answer :
Referee - It is able to manage resources through the different applications running on a physical machine. For example, an operating system can stop one program and start another.
Illusionist - It provides the illusion that each program has the computer's processors entirely to itself. For example, when we start writing a program "Hello World", we do not have to worry about how much physical memory the system has because the OS provides the illusion of nearly infinite memory.
Glue - It provides a set of common services that facilitate sharing among applications. For example, a file written by one application can be read by another application.

1.2 (2 pts) In designing an OS, there are a number of criteria to consider such as reliability, portability and performance. If you were to design an OS for a game console, out of these three criteria, what is the least important to consider? Why?

Answer :
The least significant we have to concern is portability because when we start designing OS for a game console, such as PS5 or Nintendo Switch, we want the users to use only our game consoles, and moreover some of games are exclusive games (being developed for and released only on certain platforms), so we do not want to support other platforms.

1.3 (2 pts) How does BIOS detect if a given disk is bootable?

Answer : It detects the valid bootloader in which code must have magic signature of 0xAA at byte 510 and 0x55 at byte 511.

1.4 (2 pts) Explain briefly how to print out a character in a bootloader?

Answer : we invoke the interrupt to print the character that we store characters in the register, clear the interrupt flag, and halt execution.

Problem 2: Kernel and system calls

2.1. (4 pts) What is dual-mode of operation? Why is it important?

Answer :
It is the system that keeps the system safe, by dividing control into two modes of operation,
Kernel mode and User mode because it wants to secure the safety of that system so in user mode some actions are not permitted.

2.2 (3 pts) Name THREE different events that trigger a transition from user mode to kernel mode.

Answer :

1) **We call the system calls, so the computer switches from user mode to kernel mode.**
2) **When we resume after an interrupt / processor exception / system call is done.**
3) **When we start a new process or switch to a different process.**

2.3 (3 pts) Name THREE system calls that you use in your shell project.

Answer :
exit()
fork()
exec()
wait()
open() or close() the files.

Problem 3: Processes and Signals

3.1 (2 pts) How many processes in total are created by the following program?

Answer : There are 8 processos.

3.2 (2 pts) Normally, in a shell implementation, you call fork() then exec() to start a new process. What would happen if we call exec() then fork() like the following:
What is the impact of this change to the shell, if any? (Explain)

Answer : It is not able to run exec() because we cannot start another program and get it back from it without fork(). So the program is not going to execute.

3.3 (3 pts) Name THREE different ways a process may react to a signal delivered by the kernel.

Answer : There are 3 different ways.

1) **Ignore - the signal (do nothing).**
2) **Terminate - the process (with optional core dump) .**
3) **Catch - the signal by executing a user-level function called signal handler.**

3.4 (1 pt) What is the UNIX command for sending a signal?

Answer : The kill command

3.5 (4 pts) What is the exit status of a process? How does the parent process retrieve this information?

Answer : The exit status of a process in computer programming is a small number passed from a child process to a parent process when it has finished to tell how it was terminated.
Whenever a process exits, the kernel keeps some information about it. Among this information is the return code. While the parent process has not terminated, it can call wait or waitpid to get that information.

Problem 4: Threads, Synchronization and Deadlocks

4.1 (6 pts) Consider a threaded program that is having an unknown performance issue. Give a possible cause and pick a resolution for each of the scenarios below.

Answer:

a) **Possible cause: lacking of enough threads**
   Resolution : Use more threads
b) **Possible cause: the overuse of CPU**
   Resolution : Get faster CPU
c) **Possible cause: the overuse of disk**
   Resolution : Get faster disk

4.2 (6 pts) In a social media application, it is common to have a “follow” functionality where one user is following and being followed by other users. Consider the following code:
a) When perform_follow() is called by many threads simultaneously, is it possible for the application to have a deadlock? How?
b) If so, can you suggest a modification to the code to eliminate any deadlock possibility.

Answer :

a) **It is possible to be in a deadlock, what if thread 1 and thread 2 are called at the same time, but User_id of thread 1 is higher than thread 2, since when thread 1 is locked, it is going to be stuck in the circular wait.**
b) **Before we start the process, we have to compare between those two user id, and start locking the lower user id first to avoid the circular wait.**

4.3 (2 pts) For each of the following situations, indicate whether kernel threads or user-level threads are more suitable.
a) A multi-threaded video player application where each thread reads different parts of the video from disk.
b) A multi-threaded game where each thread is rendering different part of the screen by using data cached in memory (no I/O operation needed)

Answer :
a) User-level thread
b) Kernel thread

4.4 What are the possible values for c after both threads complete? You can assume that reads and writes of the variables are atomic, and that the order of statements within each thread is preserved in the code generated by the C compiler

Answer : 4,7,14,13,-3

4: Execute thread 1 completely, then execute thread 2.
7: Interrupt thread 1 before c = b + a, and then execute thread 2, followed by executing thread 1 again.
14: Execute thread 2 till b = 10 is done, then interrupt it, and execute thread 1 completely.
13: Execute thread 2 completely, then thread 1.
-3 : thread 1, check a < 0, return 0. Switch to thread 2 and execute. Back to thread 1, and we got -3.

Problem 5: CPU Scheduling

5.1 (4 pts) A common scheduling algorithm is First-Come-First-Serve (FCFS) where jobs are assigned to CPU based on their arrival times. Describe a potential issue with FCFS algorithm.

Answer :
The most crucial problem would be what if the first-come process is taking more time than other processes, so it would take more average time. For example, assume that we have 3 processes in the order, process 1 takes 20 burst time, process 2 takes 5 burst time, process 3 takes 5 burst time. So, in total process 2 needs to wait until process 1 finishes (20 burst time), also process 3 needs to wait until process 2 finishes (20+2 burst time), it could be a very long time.

5.2 (4 pts) In Shortest-Remaining-Time-First (SRTF) scheduling, the algorithm picks a job with the smallest amount of time remaining to schedule first. In practice, it is difficult to know the remaining time of a process in advance. What can the OS do to approximate the remaining time of a given process?

Answer :
It is hard to know the actual length of time of that process, but we can estimate the length of the next cpu burst.
Basically, we use the length of previous CPU bursts to determine the next CPU burst, using exponential averaging. Then since we already determined the next CPU burst,assume that now we run the 1st process with 20 CPU burst time and the 2nd process arrive at arrival time 5 with 5 cpu burst time, by comparing the arrival time and burst time with those two, the algorithm pause the 1st process and get the 2nd process done, then resume on running the 1st process
(this one is really hard to explain without any image).

5.3 (3 pts) In round-robin scheduling, it is common to use a fixed-sized quantum.
a) What is a quantum?
b) What is a benefit of using a small quantum?
c) What is a benefit of using a large quantum?

Answer :
Quantum is a fixed amount of time that Os allows the process to run and then it switches to another process. For example, if the quantum time is 5, the 1st process takes 20 CPU burst time to finish, the 2nd process takes 10 CPU burst time to finish. The process starts with the 1st process, it runs 5 CPU burst time on the 1st process, and switches to the 2nd process, and switches back to 1st process, keep doing this until all the processes are done.

Small Quantum = the response time of the processes is too much which may not be tolerated in an interactive environment.
Large Quantum = it causes unnecessarily frequent context switches leading to more overheads resulting in less throughput.

Problem 6: Protection and Security

6.1 (4 pts) Explain what is the principle of least privilege. Give a concrete example exercising the principle of least privilege. You may use non-computer related examples.

Answer :
Programs, users and systems should be given just enough privileges to perform their tasks.
So the principle of least privilege is the permissions that can limit damage if an entity has a bug, gets abused.
For example, when using Google Maps in an Iphone, the system needs you to allow this application to determine and share your location, but we should not give them too much personal information, otherwise it can be harmful to ourselves.

6.2 (4 pts) For each of the following, indicate whether an access list or a capability list is being used and argue the weakness of the method.
a) During the covid-19 situation, many buildings set up temperature screening stations at their entrances. Those who already pass the screening will receive a sticker and can enter the building.
b) Every time someone wants to enter the Science Division, he or she must tap your ID card on the scanner

Answer :

a) **Capability list - weakness is that evocation capabilities can be inefficient. In this case**
b) **Access list - weakness is that Determining set of access rights for domain non-localized is so difficult and every access to an object must be checked in which can be very slow. In this case, imagine if your organization is really huge, it could take a bit of time to access.**