Unit 4 Sample Problems - Operating-System Structures

In this sample problem set, we will practice concepts of operating system structures.

- Length: 50 minutes with discussion.
- Questions: Q1-Q3, Q6-Q7 (optional: Q4-Q5, Q8)

1 Operating-System Services
2 User and Operating System Interface
3 System Calls

1. [Karaliova] Consider the following pieces of functionality:

   (a) Finding duplicate nodes in a linked list that is loaded in memory

   (b) Providing a report on system health that details all processes currently running on a system

   (c) Compressing an image after it is loaded from a file

   Which of these would need to be implemented as a system call?

2. [Karaliova] Consider the following function from C stdio.h library that handles renaming of a file:

   ```
   int rename (const char *oldname, const char *newname)
   ```

   What conditions/events does the underlying system call need to check for in order to rename a file? Name at least three. [Hint: think in terms of exceptions that can be thrown while performing this operation].

3. [Karaliova] For most programming languages, the run-time support library provides a system call interface that translates function calls into corresponding system calls. Why do we need an intermediate between the two instead of invoking system calls from a program directly? Explain.

1 Types of Systems Calls

4. [Karaliova] One of Unix process control type system calls is fork(). Fork() creates a new process by duplicating the calling process. The new process is referred to as the child process, the calling process is referred to as the parent process. The child process and the parent process run in separate memory spaces. Using fork() or any other example of system call of process control type (Unix or Windows), explain why it is important for programming APIs to have access to system calls.

2 Operating-System Design and Implementation

5. [Lisonbee] Why are operating systems generally implemented using lower level languages? For instance, Windows, MacOS, and Linux are all implemented in mostly C++, C, and some assembly. **Explain** why these types of languages are chosen over higher level ones such as Java or Python.

3 Operating-System Structure

6. [Acuña] Say that you are designing an operating system for an interstellar probe that is meant to run for hundreds of years. What structure (simple, layered, microkernel, or modular) should you choose for its kernel? **Explain.**

7. [Lisonbee] Compare and contrast the organic and microkernel operating system structures. **Explain** how the efficiency (speed), security, and maintainability of each compare, and why.

8. [Lisonbee] Different operating system structures offer both benefits and drawbacks over one another, and it's important to understand what kind of structure should be used for different use-cases.

   Consider the following situation. You are tasked with creating a very secure piece of software for an embedded system that should have a relatively small amount of overhead. Given the nature of embedded systems, once you deploy the system you won't be able to update the software, thus maintainability isn't a large factor.

   What structure (simple, layered, microkernel, or modular) should you choose for its kernel? **Explain**.