HOMEWORK 2

---

1. What is the purpose of system calls?

System calls serve as the essential bridge between user applications and the operating system, enabling programs to request crucial services such as file management, memory allocation, and process control. Without system calls, applications wouldn't have direct access to hardware or system resources, as the operating system maintains control to ensure security and stability. For example, when a program needs to read a file, it doesn't directly access the hard drive; instead, it makes a system call, which the operating system handles to retrieve the necessary data safely.

These calls act like translators, allowing software to interact with low-level system operations without requiring developers to manage the complex details of hardware communication. Common system calls include opening files, creating processes, allocating memory, and handling network communication. They also ensure that different applications can run simultaneously without interfering with each other, maintaining system efficiency and preventing crashes.

By providing a structured way to access system resources, system calls enhance security, as they prevent unauthorized access to critical functions. They also improve performance by optimizing resource management. In essence, system calls are the backbone of any operating system, ensuring smooth communication between software and hardware while maintaining stability and security.

2. List five services provided by an operating system and explain how each creates convenience for users. In which cases would it be impossible for user-level programs to provide these services? Explain your answer.

Five OS Services and Their User Convenience

1. **Process Management**
    1. Manages CPU scheduling and multitasking.
    2. **Convenience**: Runs multiple programs smoothly.
    3. **Limitation**: User programs can't allocate CPU time efficiently.
2. **Memory Management**
    1. Allocates and frees RAM for processes.

    2. **Convenience**: Prevents memory conflicts and wastage.
    3. **Limitation**: Programs can't directly manage memory safely.

3. **File System Management**
    1. Organizes and controls file storage.
    2. **Convenience**: Easy file access and management.
    3. **Limitation**: Programs lack uniform file handling.

4. **Device Management**
    1. Controls hardware interactions.
    2. **Convenience**: Plug-and-play device support.
    3. **Limitation**: Programs can't directly access hardware.

5. **Security & Access Control**
    1. Enforces authentication and protection.
    2. **Convenience**: Prevents unauthorized access.
    3. **Limitation**: Programs can't enforce security effectively.

Why Can't User Programs Provide These?
They lack direct hardware access, scheduling control, and security enforcement, making OS intervention essential.

3. What is the main advantage of the layered approach to system design? What are the disadvantages of the layered approach?

Main Advantages:

1. **Modularity** – The system is divided into well-defined layers, making it easier to design, implement, and modify.
2. **Ease of Maintenance** – Since each layer operates independently, changes in one layer do not affect others, simplifying updates and troubleshooting.
3. **Enhanced Security** – Lower layers handle core hardware and system operations, preventing unauthorized access or direct system manipulation.
4. **Better Organization** – Each layer has a specific function, leading to a structured and logical system design.
5. **Scalability** – New features or hardware can be added by modifying or introducing new layers without disrupting the entire system.

Disadvantages:

1. **Performance Overhead** – Layered abstraction increases processing time as requests must pass through multiple layers.
2. **Complex Implementation** – Designing and maintaining well-defined layers requires careful planning, which can be challenging.
3. **Restricted Flexibility** – Higher layers depend on lower layers, limiting direct access to hardware, which may reduce efficiency in real-time applications.
4. **Difficult Debugging** – Errors in lower layers can be hard to trace since they affect multiple higher layers.

Despite its drawbacks, the **layered approach is widely used** in operating systems because of its **modularity, maintainability, and security**, making it a preferred design method.

4. Describe three general methods for passing parameters to the operating system.

When a program makes a system call, it needs to pass parameters to the operating system. There are three common methods to achieve this:

1. **Passing Parameters in Registers**
    1. The simplest method involves storing parameters directly in the CPU registers before making a system call.
    2. **Advantage**: Fast and efficient, as registers provide quick access.
    3. **Disadvantage**: Limited by the number of available registers, restricting the number of parameters that can be passed.
2. **Using a Memory Block (Table or Buffer)**
    1. Parameters are stored in a designated memory block, and the address of this block is passed to the operating system via a register.
    2. **Advantage**: Allows passing large amounts of data efficiently.
    3. **Disadvantage**: Requires additional memory operations, which may slow down execution.
3. **Pushing Parameters onto the Stack**
    1. Parameters are placed onto the program's stack, and the OS retrieves them during execution.

2. **Advantage**: Flexible, as it allows passing a variable number of parameters.
3. **Disadvantage**: Stack management adds overhead, and incorrect handling may lead to security risks.

Each method has trade-offs between speed, flexibility, and memory usage. The choice depends on the OS design and the type of system call.

5. How are iOS and Android similar? How are they different?

Similarities Between iOS and Android

1. **Smartphone Operating Systems** – Both power mobile devices with app-based interfaces.
2. **App Store Ecosystem** – iOS uses the App Store, while Android uses Google Play.
3. **Security Features** – Both offer biometric authentication, encryption, and app sandboxing.
4. **Multitasking & Notifications** – Support background apps, push notifications, and quick toggles.
5. **Regular Updates** – Receive frequent software and security updates.

Differences Between iOS and Android

1. **Customization** – Android allows themes, widgets, and third-party launchers; iOS has limited customization.
2. **App Distribution** – iOS apps are restricted to the App Store, while Android supports sideloading APKs.
3. **Device Variety** – Android runs on multiple brands (Samsung, Google, OnePlus); iOS is exclusive to Apple.
4. **File Management** – Android allows external storage and open file access; iOS has a closed file system.
5. **Integration** – iOS works seamlessly with Apple's ecosystem; Android is more flexible across devices.

Conclusion

- **iOS** offers a secure, seamless experience with tight ecosystem integration.
- **Android** provides more customization, flexibility, and device choices.