HW#2 – Jiading Zhou

**1. What is the purpose of system calls?**

System calls serve as the interface between user applications and the operating system kernel. They allow programs to request services—such as process creation, file manipulation, device control, and inter-process communication—that require direct interaction with hardware or privileged operations. By transitioning from user mode to kernel mode, system calls ensure that sensitive operations are executed securely and correctly, protecting system 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.**

**Program Execution:**

* *Convenience:* The OS loads programs into memory, sets up the execution environment, and manages the running processes (including context switching and scheduling). This abstraction spares users and application developers from having to manage hardware details directly.
* *Why not user-level:* Direct management of process creation, scheduling, and termination requires privileged access to CPU and memory management hardware, which is not available to user programs.

**I/O Operations:**

* *Convenience:* The OS provides uniform interfaces for interacting with a variety of input/output devices (keyboards, displays, disks, etc.), abstracting the hardware differences and complexities.
* *Why not user-level:* Handling diverse hardware protocols and ensuring safe, efficient communication with devices require privileged instructions and low-level hardware access.

**File System Manipulation:**

* *Convenience:* Users can create, delete, read, and write files without needing to know the details of the underlying disk structure or storage mechanisms. The OS manages file organization, access permissions, and data integrity.
* *Why not user-level:* Implementing robust and secure file management necessitates direct control over storage hardware and careful coordination to prevent data corruption, which is only feasible in kernel mode.

**Communication:**

* *Convenience:* The OS provides mechanisms (such as pipes, sockets, and shared memory) for processes to exchange data, enabling functionalities like networking and inter-process communication.
* *Why not user-level:* Secure and efficient communication often involves managing network protocols and hardware interrupts—tasks that require the elevated privileges available only to the operating system.

**Resource Allocation and Management:**

* *Convenience:* The OS allocates system resources (CPU time, memory, and peripheral access) among competing processes, ensuring fairness and efficiency without user intervention.
* *Why not user-level:* Coordinating resource allocation and preventing conflicts between processes require a centralized authority with full control over hardware resources, which user-level programs lack.

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

**Advantage:**

* **Modularity and Abstraction:**  
  The layered approach divides the operating system into a hierarchy of layers, each built upon lower layers. This modularity simplifies design, debugging, and maintenance since each layer handles a specific set of functionalities and communicates only with its adjacent layers.

**Disadvantages:**

* **Performance Overhead:**  
  Crossing multiple layers can introduce additional overhead due to extra abstraction and context switching, potentially reducing performance.
* **Rigid Structure:**  
  Strict layering may force certain operations to go through unnecessary layers, limiting optimization and direct access that might be more efficient in an integrated design.
* **Inflexibility:**  
  It can be difficult to change the functionality of a lower layer without affecting the entire system, leading to challenges in adapting or extending the system.

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

**Using Registers:**

* **Method:** Parameters are placed in designated CPU registers before invoking the system call.
* **Pros/Cons:** This method is fast because registers are directly accessible, but it is limited by the number of available registers and the size of data that can be passed.

**Using Memory (Block/Buffer):**

* **Method:** The calling process places parameters in a contiguous block of memory (such as a structure or an array) and passes a pointer to this block in a register.
* **Pros/Cons:** This method allows the transfer of larger amounts of data, though it requires careful handling to ensure security (e.g., validating the memory block) and correct data formatting.

**Using the Stack:**

* **Method:** Parameters are pushed onto the process stack prior to the system call. The kernel then accesses these parameters from the stack.
* **Pros/Cons:** This method is common in high-level language function calls and offers flexibility, but it might involve additional overhead due to stack management.

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

**Similarities:**

* **Mobile Operating Systems:** Both are designed primarily for smartphones and tablets, providing support for multitasking, memory management, and a rich graphical user interface.
* **App Ecosystems:** They each offer extensive application frameworks (iOS uses Cocoa Touch with Swift/Objective-C; Android uses the Android SDK with Java/Kotlin) and have dedicated app stores (Apple App Store and Google Play Store).
* **Unix-like Foundations:** iOS is built on the XNU kernel (which has Unix-like characteristics) and Android is built on a modified Linux kernel, ensuring robust multitasking and security features.

**Differences:**

* **Source Model and Customizability:**
  + *iOS:* A closed-source, proprietary OS with tight integration between hardware and software, leading to a more controlled and uniform user experience.
  + *Android:* Based on the open-source AOSP, allowing for greater customization and a broader range of hardware implementations by various manufacturers.
* **User Interface and Ecosystem Control:**
  + *iOS:* Known for its strict app review process and curated ecosystem, which can lead to a more secure but less flexible environment.
  + *Android:* Offers more flexibility and customization options, though this can sometimes result in a more fragmented ecosystem.
* **Hardware Integration:**
  + *iOS:* Exclusively runs on Apple devices, enabling a tightly integrated hardware-software experience.
  + *Android:* Runs on a wide variety of devices from different manufacturers, which means a more diverse range of hardware capabilities and user experiences.