Homework 1

Yash Shah

February 4, 2026

Question 1

How does the distinction between kernel mode and user mode function as rudimentary form of protection? Give one real-world example (e.g., buffer overflow, privilege escalation) where this distinction either prevented or failed to prevent an attack.

Ans.

The distinction between kernel mode and user mode functions as a rudimentary form of protection by enforcing a hardware-level separation between trusted operating system code and untrusted application code. User-mode programs are prevented from executing privileged instructions, directly accessing hardware, or reading and writing kernel memory; any attempt to do so triggers a CPU exception and terminates the process. A real-world example illustrating both the effectiveness and limitations of this protection is privilege escalation via kernel vulnerabilities: while a typical user- space buffer overflow cannot directly modify kernel data due to user-mode restrictions, flaws in kernel code (such as race conditions in memory management) can be exploited to bypass this boundary.

Question 2

Explain the difference between a device controller with just a local buffer and a device controller with a local buffer that supports Direct Memory Access (DMA). Discuss below points:

- How data is transferred
- How much CPU time is consumed
- Which method is more efficient and why

Ans.

A device controller with only a local buffer transfers data by relying heavily on the CPU to move each unit of data between the device's local buffer and main memory. In this model, the device controller signals the CPU (often via interrupts) when data is ready, and the CPU then executes instructions to copy the data from the controller's buffer into memory or vice versa. As a result, a significant amount of CPU time is consumed managing these data transfers, since the CPU is involved in every movement of data. In contrast, a device controller that supports Direct Memory Access (DMA) allows data to be transferred directly between the device's local buffer and main memory without continuous CPU intervention. The CPU's role is limited to initializing the DMA transfer by providing parameters such as the memory address, transfer size, and direction, after which the DMA controller handles the data movement independently and interrupts the CPU only upon completion.

Question 3

Find different levels of cache available in your PC/laptop. Write the space allocated to each cache. Paste a snapshot of the cache information of your system. Explain very briefly what the role of different levels of cache in the system.

Ans.

My laptop uses a multi-level cache hierarchy consisting of L1 and L2 caches. The L1 data cache is 65,536 bytes (64 KB) and the L1 instruction cache is 131,072 bytes (128 KB), making the L1 cache the smallest and fastest level, located closest to each CPU core for immediate access to frequently used data and instructions. The L2 cache is 4,194,304 bytes (4 MB) and serves as a larger, slightly slower cache that stores data likely to be reused, reducing the need to access main memory.

Question 4

The services and functions provided by an operating system can be divided into two main categories. Briefly describe these two categories, then give one example of each from your own usage (e.g., file backup, running Zoom, compiling code). Why is it important that these categories are separated?

Ans.

The services and functions provided by an operating system are commonly divided into user-oriented services and system-oriented services. User-oriented services are designed to make the computer system convenient and efficient for users and applications; they include functions such as program execution, I/O operations, file- system manipulation, and communication. An example from my own usage is running a web browser, where the operating system loads the program into memory, schedules CPU time, and handles input and output seamlessly. In contrast, system-oriented services focus on the internal operation and efficiency of the system itself; these include resource allocation, accounting, protection, and security. An example of this from my usage is permission enforcement, where the operating system manages disk space, tracks resource usage, and ensures that only authorized processes can access certain files. Separating these two categories is important because it allows the operating system to provide a clean and simple interface to users and applications while still maintaining strict control over hardware resources and system integrity.