For each of the following questions, please write a one to two paragraph answer explaining the related concepts to the best of your ability.
1. The kernel is a collection of code used to perform core operating system functions. How does this code start running? What mechanisms are used to ensure that code written by users cannot use the kernel code in malicious ways?
2. The `fork` system call returns twice, once in the parent process and once in the child process. How is fork implemented in the kernel generally, and specifically, how does it return multiple times?
3. ASLR, or Address Space Layout Randomization, is a security technique used in most modern systems where the operating system randomly arranges the memory space for a process, offsetting the text, stack, heap, and shared libraries each by some random amount every time a process is loaded. What does this protect against and why is it effective?
4. Multi-tab web browsers are a classically parallel program. Originally they were multi-threaded, but in recent years they have switched to be multi-process. Explain what about web browsers makes them a good fit for parallelization, and the relative advantages and disadvantages to threads and process. Speculate on why they have switched over to processes from threads.
5. Python’s Global Interpreter Lock, or GIL, prevents multiple threads from running in the python interpreter at once. What are the implications for multi-threaded programs in python? In what situations will multi-threading still increase program speed, and in what situations will the GIL prevent speed-ups from multi-threading?
6. All synchronization primitives are based on the test and set instruction, which writes 1 to a memory location, and returns the previous value at that location (either 1 or 0), as an atomic operation. Explain how you can construct a semaphore from test and set, and how you can construct a lock from a semaphore.
7. The x86-64 architecture uses a 4 level page table. What are the advantages to having a page table with this many levels? How does this not cause all memory references to be incredibly slow?
8. A cache can be indexed and tagged using either physical or virtual addresses, giving us Virtually Indexed/Virtually Tagged, Physically Indexed/Physically Tagged, and Virtually Indexed/Physically Tagged caches. Explain the relative advantages and disadvantages of each of these systems, speculate on why Intel uses a Virtually Indexed/Physically Tagged cache, and explain why there are no Physically Indexed/Virtually Tagged caches.
9. Explain how Fork uses the Copy on Write mechanism to save space in memory.
10. Historically the operating system controlled exactly where it wrote a sector to a hard disk, addressing blocks by cylinder, head, and sector. Modern operating systems instead use Logical Block Addressing, where they treat the disk as an array of blocks. Why has this changed? What are the advantages of the OS not controlling the physical location of a block?
11. Unix uses the file abstraction for almost all I/O devices. What are some advantages and disadvantages of treating devices as file?
12. Explain how and why Unix uses both Access Control Lists and Capabilities for files.