CSE 221 -- Graduate Operating Systems

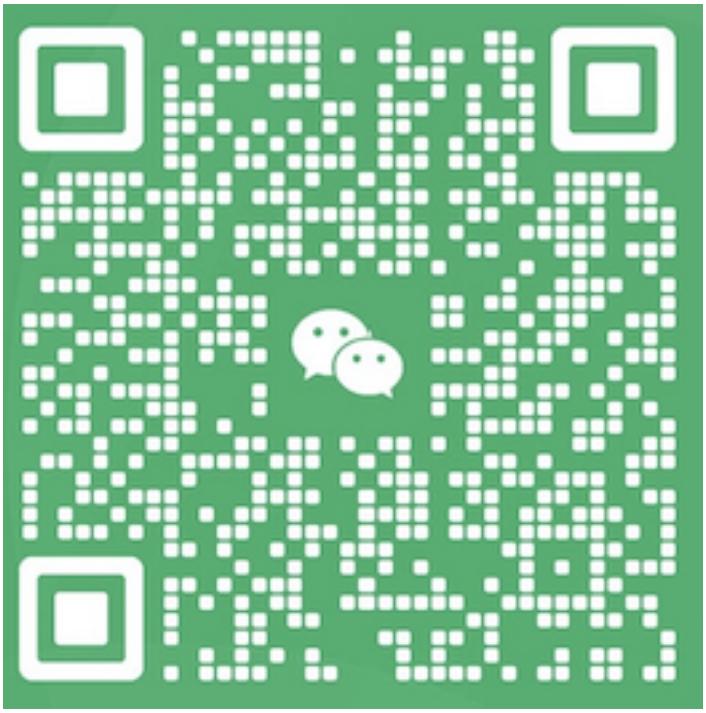
cseweb.ucsd.edu/classes/wi22/cse221-a/hw2.html

CSE 221: Homework 2

Due Thursday, February 24 at 11:59pm

Answer the following questions. For questions asking for short answers, there may not necessarily be a "right" answer, although some answers may be more compelling and/or much easier to justify. But I am interested in your explanation (the "why") as much as the answer itself. Also, do not use shorthand: write your answers using complete sentences.

When grading homeworks, we will grade one question in detail and assign full credit for technical answers to the others.



1. In contrast to blocking locks as provided by semaphores and monitors, a different form of synchronization is known as "non-blocking" or "wait-free" synchronization. With non-blocking synchronization, threads accessing a shared data structure do not block each other. (Linux uses a related kind of synchronization known as read-copy update (RCU) locks, which can have very efficient implementations.)
   The general idea is that a shared data structure is accessed via a single shared pointer. Threads just reading the data structure do not acquire a lock and are not blocked by other readers or writers. They happily use the value of that pointer to read data from the data structure. Threads that write (update) the data structure follow these steps:

   1. Allocate a separate, private instance of the data structure.
   2. Copy the old, shared version of the data structure into the new, private instance (copying just requires reads from the shared version).
   3. Update (write to) the new, private instance as needed.
   4. Atomically change the pointer referencing the data structure to point to the new instance instead of the old. Any previous reader threads referencing the old data st armed. Any new reader thre e and see the updates fr
      This handl ple writers, then the last ste goes to update the shared poi n modified since it made a co lue of the shared pointer bef ack to step 2.

   Assuming these brief explanation.

   a. What kinds eads) work particularly
   b. Does this k nds of data structures,
   c. Can reade
   d. Can writers starve readers?
   e. Can readers starve writers?
   f. Can writers starve writers?
   g. Can deadlock occur?
   h. Is it safe for a thread to be arbitrarily suspended or stopped while it is accessing a shared data structure?

2. Exokernel and L4 represent approaches for providing protection and extensibility. Xen represents an approach for providing virtualization and isolation (or, alternately, is an extreme version of extensibility since it goes even beyond Exokernel in exposing the hardware interface to unprivileged code). Consider a Web server as a motivating application-level service running on each of these three system structures, each hosting the OS described in the paper.
   For each of the three systems, consider the path a network packet containing an HTTP request takes as it travels from the network interface card to a Web server process running at user level:

   a. Identify the various protection domains in the system for this scenario. Which domains are privileged, and which are unprivileged? (Feel free to draw "boxes-and-kernel-boundary" diagrams if you find them helpful.)
   For example, if the system were standard monolithic Linux, the protection domains would be the kernel and the Web server process with its address space. The kernel is privileged, and the server process unprivileged.

   b. Describe t … hrough the protection … main crossing, state the c … cross protection domains.

   c. Argue whic … performance Web service wit … the Web server code into t … fy your argument and be su…

   d. Further co … ault on a page in its address … pagation of the page fault … es the page fault? Whose po … ault?
   For examp … the CPU would raise an in … or to a Linux kernel interrupt h … ld allocate a physical page … the page table entry with the valid mapping. The Linux kernel would then return from the interrupt.

3. The notion of "thin" clients has been an attractive idea across the decades. We have seen a few research systems designed to accomodate diskless workstations in the 1980s (V, Sprite, Plan 9). Machines like X Terminals and Sun Rays reached commercial status in the 1990s and 2000s, and Chromebooks perhaps represent the closest thing to a thin client in the modern era. Yet, thin clients have not revolutionized computing. Most of the client devices that we continue to use today (desktops, laptops, smartphones) are full-feature computers still running timesharing operating systems.
    a. First argue why thin clients were seen as a compelling computing platform, and justify your argument with supporting claims.
    b. Then argue why thin clients have not been widely adopted as a platform, also justifying your argument.
    c. Which factors in favor of thin clients are still valid today?