CS33 Homework Assignment 8 Solutions

Fall 2020

1. Virtual memory is a mechanism by which each process may run in a separate address space.
    a. Describe circumstances in which identical virtual addresses of different processes are mapped to different real addresses.

       Answer: This is what happens in the normal case of two independent processes, each running in a separate address space. Assuming they are not sharing their text sections (i.e., they're running different programs), their virtual addresses are all mapped to different real addresses.

    b. Describe circumstances (i.e., how you might make this happen) in which different virtual addresses of different processes are mapped to the same real address.

       Answer: This could happen through the use of *mmap*: two processes map the same file into different locations within their address spaces. A particular block of the file that resides in memory will be mapped into by a different virtual address of each process.

    c. Describe circumstances (i.e., how you might make this happen) in which different virtual addresses of one process are mapped to the same real address.

       Answer: As in part b, except that one file is mapped into one process, using *mmap*, twice, into two different portions of the address space.

2. Modern processors cache recently (and soon-to-be) accessed instructions and data in high-speed memory composed of SRAM (static RAM, which is much faster than dynamic RAM). Suppose you are a computer architect and are designing such a cache system. One concern is how items are identified in the cache: either using virtual addresses (as generated by the program) or real addresses (after being translated by the hardware from virtual addresses). One concern, for the case of using virtual addresses, is distinguishing instructions or data of different processes that are at the same virtual address but are, in fact, different. This might be done by associating with each process a unique *address-space ID* (ASID) and tagging items it puts into the cache with its ASID. Further complicating things is that, while each core has its own L1 and L2 caches, all cores on a chip share one L3 cache.

   Which would be the most reasonable choice: using a cache accessed using virtual addresses (plus ASIDs) or a cache accessed using real addresses? Explain. [Hint: keep problem 1 in mind.]

   Answer: Though both approaches have been used, assuming that the situations described in 1b and 1c are possible in the system (which they would be if the system is a reasonably modern version of Unix), then the only reasonable choice would be to access the cache using real addresses. Otherwise the system could not determine that two virtual addresses refer to the same real address.