CSE 430 Summer 2014 - Homework 1

Ryan Dougherty

Question 1.13

The issue of resource utilization shows up in different forms in different types of operating systems. List what resources must be managed carefully in the following settings:

(a) Mainframe or minicomputer systems: One would need to manage CPU, memory resources, and storage carefully because he/she would not want any one user to take more than his/her fair share of resources. Also, one needs to manage bandwith over the network carefully because other users are accessing the same computer/server/mainframe, and the more users that are using the same machine, the less throughput is possible per user.

(b) Workstations connected to servers: One would need to manage CPU and memory resources carefully because of the same reason as mainframe and minicomputer systems: so that no one user takes more than his/her fair share of resources.

(c) Mobile computers One would need to manage power consumption (because of the screen, CPU, etc.) and usage of memory resources carefully because this kind of device has limited resources, such as the battery, whereas desktops and similar machines do not have that restriction.

Question 1.22

Many SMP systems have different levels of caches; one level is local to each processing core, and another level is shared among all processing cores. Why are caching systems designed this way? SMP systems are designed this way to focus on speed of access to the cache, and the cache's locality to the CPU. If the cache on the device is physically closer to the CPU, that cache is faster to access, mainly because of locality reasons. But, as fast caches get larger, so does the cost. The best way, therefore, to approach this problem is to provide a few small, fast caches, and large, slower caches (i.e. shared among different processors). This is why SMP systems and their caches are designed in this manner.

Question 1.23

Consider an SMP system similar to the one shown in Figure 1.6. Illustrate with an example how data residing in memory could in fact have a different value in each of the local caches. Let S be an SMP system, and let P_0 and P_1 be 2 processors in S. An example of how data in memory could have a different value in each of the local caches is the following:

(1) P_0 reads datum D with some value from memory into its cache,

(2) P_1 does the same as P_0,

(3) P_0 updates value of D.

P_0's updating of D updates the value in the shared memory, but since P_1 has the datum in its local cache, P_1 does not receive the update that P_0 made in Step 3.

Question 1.25

Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs.Enforcing memory protection can be done by keeping a record on the bounds of what memory is legal to access by the process. How the bounds can be stored could be in some number of registers (for low, high bounds) or in a cache/main memory. Legality of access can be checked every time a memory location is accessed by comparing the accessed location to see if the location is within the acceptable bounds (by comparison).

Question 2.5

What is the purpose of the command interpreter? Why is it usually separate from the kernel? The command interpreter reads user commands or commands from a file (and then executes them). If read from a file, the command interpreter converts them into system calls. It is usually separate from the kernel because the command interpreter can possibly be changed.

Question 2.17

Would it be possible for the user to develop a new command interpreter using the system-call interface provided by the operating system? It is possible for a user to develop a new command interpreter using the system-call interface provided by the OS. This can be done by using the API for system calls given by the OS. The purpose of the command interpreter (given in Question 2.5 above) is that it reads user commands or commands from a file (and then executes them). The user will just have to forward any input calls to the matching API provided by the OS (or system calls directly, etc.). Since all of these functionalities are accessible at user-level, developing a new command interpreter in this fashion is entirely possible.

Question 2.18

What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches? The two models of interprocess communication are the: (1) message-passing model, and (2) shared-memory model.

- The strength of the message-passing model is that it is easier to implement this model than the shared-memory model.
- The strength of the shared-memory model is communication is allowed to be done at memory speed, and therefore is faster than the message-passing model (when on only one machine).
- The weakness of the message-passing model is that it is only useful for exchanging small amounts of data, because it is slower than the shared-memory model (connection setup time is longer).
- The weaknesses of the shared-memory model are (1) synchronization issues involving different processes simultaneously updating a shared memory location, and (2) possible problems involving processes (attempting to) access invalid memory locations.