**Practice 1 Solutions:**

**Chapter 1, Problem 4**

(a) If everyone is honest and trustworthy, then we should trust the requested amount of work to complete each job/task.

We should not give the entire processor to each application until it no longer needs it. Time-sharing is suggested: neither the longest-first nor the shortest-first approach works best since it may starve other applications.

(b) The OS should perform dynamic loading/unloading where portions of the image of each application is loaded in the physical memory from time to time to make sure all applications are served at the same time.

(c) It depends on whether each user has a quota for the disk. If not, a user should get as much as what's available. Otherwise, the user can only get a certain size up to the limit. When disk space is limited, setting quota is a common practice.

**Chapter 1, Problem 9**

Crash consistency is a real challenge. System developers must understand the system crash behavior and an appropriate protocol to update the system in case of crash. Such protocol includes recording the intended update and then using logs to find out at which step the update halts due to system crash. Then, either a partial or full update redo of the sequences of intended updates after that crash time point needs to be performed to ensure consistency.

**Chapter 2, Problem 1**

The user stack pointer may be corrupted. Switching to the kernel stackensures that there is a valid memory region to store the process state.

**Chapter 2, Problem 7**

Page 72

(a) the instruction is used by the *handler software*

(b) restore the code segment, program counter, execution flags, stack segment, and stack pointer from the kernel's interrupt stack