a. Signal - voltage that represents coded information
b. Interrupt - signal that stops the main function and figures out what to do next
c. Interrupt vector – list comprising locations of various interrupt handlers
d. Trap - aka exception – interrupt caused by an exceptional condition (breakpoint, division by zero etc..)
e. Process - instance of a computer program
f. Context switch - procedure the cpu follows to switch from 1 task to another, neither conflict with e/o
g. Zombie – computer connected to the internet that has been compromised by a hacker.
h. Exec() - similar to fork – starts a new process
i. Quantum - individual unit of data

2. If an I/O-bound application typically exhibits a low degree of locality, what effect is this likely to have on its overall performance?

   Little to no decrease because programs waiting for the cpu

3. Why would modern caches index virtual rather than physical addresses?

   Virtual address space doesn't need to be physically contiguous

4. Give an example of a typical fault arising from a memory reference. Is it necessarily a bad thing?

   you can force a page fault to cause a program to assign more virtual memory to a program

5. What is the difference between a process and a thread?

   Processes are completely independent of each other unless the OS specifically handles interaction, threads share the same chunk of memory and communicate directly.

6. When fork() returns, how do you know whether you are the child or parent process?

   Childs pid = 0; Parents pid = positive non zero #

7. Does an average single-threaded user program need to worry about concurrency? Why or why not?

   Concurrency handles a system that has multiple processes using the same piece of data, so in a single threaded machine there is no need for concurrency.

8. Why arrange the processes in a hierarchy? Why not just make them all "equal"?

   Easier for the user the understand a hierarchy when you need to set priorities

9. Why would the default action upon receiving a signal be program termination?

   Program termination is the safest way to handle an unexpected signal

10. Evaluate using signals rather than global variables to control program behavior.

   Signals are expensive, so unless your running a large program global variables will keep performance from deterring.