CpSc 360, Section 2, Fall 2008, Quiz #5

Name:_____ Clemson ID:_____

Question #1 (20 points):
Are the following statements **true** or **false**?

1. All threads within a process share the same global memory. (TRUE)
2. The OS will retain the thread ID and exit status of a joinable thread after its termination until another thread calls pthread_join(). (TRUE)
3. A function running in multi-thread environment cannot use static variables. (FALSE)
4. A thread will never terminate if it does not call pthread_exit(). (FALSE)
5. Broadcasting requires datagram transport such as UDP or raw IP while multicasting can work with TCP. (FALSE)
6. IPv4 uses Class D addresses, in range 224.0.0.0 through 239.255.255.255, as the multicast addresses. (TRUE)
7. All targeted hosts in the network will retrieve broadcast datagrams from their network interfaces. (TRUE)
8. Daemon processes do not have I/O (Input or Output). (FALSE)
9. Servers started by inetd must call getpeername() to know the client's address. (TRUE)
10. Servers with frequent requests are typically not run through inetd. (TRUE)

Question #2 (20 points):
Briefly describe how to write a multi-thread program.

- Write a function to be executed by the thread. When function returns the thread is expected to terminate. This function must only have one parameter which is a pointer to a void.
- Define a structure to put all your parameters in it so that a single pointer to this data structure can be used as the parameter passed to the function.
- Call pthread_create() with a pointer to the function and a pointer to the structure as two of the four parameters. In addition, you can also pass in a pointer to pthread_t variable to get the thread ID back and pass in a pointer to an attribute structure. This parameter is usually set to be NULL.
- The thread may be terminated by calling pthread_exit() within the thread function.

Question #3 (10 points):
How does the main thread obtain the IDs of the child threads? How does a thread obtain its own thread ID?

The first parameter of pthread_create() should be a pointer to a variable of type pthread_t. The ID of the child thread will be returned by this variable.

To obtain its own ID, a thread can call pthread_self().

Question #4 (10 points):
Given the following information about a subnet, please list the limited broadcast IP address and the subnet directed broadcast IP address.

Subnet Mask is: 255.255.252.0
One IP address is: 165.91.216.50

The limited broadcast IP address is 255.255.255.255.
The subnet directed broadcast IP address is 165.91.219.255

Question #5 (20 points):
Why is multicast more efficient than broadcast? What are their differences in terms of programming?

Broadcast data package will be examined by the datalink layer, IP layer, and UDP layer before it is discarded if the host does not listen to the broadcasting. The multicast data package will be rejected by the datalink layer by hardware filtering if the host does not join the IP multicast group.

A broadcast send program needs to enable the broadcast socket option SO_BROADCAST by calling setsockopt(). It also needs to send the data to a proper broadcast IP address and a port number. A broadcast receive program can simply bind to any interface in a host and receive the broadcast package from the specific port.

A multicast send program needs only to send the data to a proper multicast IP address (Class D address) and any port number. It may set the multicast TTL value by calling setsockopt(). A multicast receive program needs to explicitly join that multicast group by calling setsockopt() to receive the multicast data. It also needs to listen to the specific port.

Question #6 (20 points):
Briefly describe how to create a daemon process.

- Fork off the parent process. (call fork())
- If we got a good PID, then we can exit the parent process.
- Change the file mode mask.
- Open necessary logs.
- Create a new SID for the child process (setsid())
- We may need to call fork() one more time here.
- Change the current working directory.
- Close out the standard file descriptors.