Byron Laferriere

IT-365

Professor Hardikar

Module Three ‘Run The Code’ Questions

*Challenge Question One:*

*What are the benefits and the disadvantages of each of the following? Consider both the system level and the programmer level.*

1. *Synchronous and asynchronous communication*

The main advantage that comes with synchronous communication is that it opens the ability for rendezvous, which is where both send() and receive() are blocking between the sender and receiver (Silberschatz et al, 2009). When it comes to the disadvantages of synchronous communication, if communication between the sender and the receiver isn’t required then synchronous mode is rendered useless and the message delivery is required to be delivered asynchronously. Because of these downfalls, most systems provide both types of communication.

1. *Automatic and explicit buffering*

Automatic buffering supplies a queue that possesses an indefinite length, which enables the sender to never have to block while waiting to copy messages. The downfall to this scheme is that there is no blueprint on how automatic buffering should be performed, which can waste extremely large amounts of memory due to the endless length. This is the pro to explicit buffering. With a predefined buffer length or size, the buffer may be blocked while waiting, but it wastes much less memory.

1. *Send by copy and send by reference*

Send by reference allows the receiver to alter the parameters and send by copy does not allow this. Programmers can use send by reference to write distributed versions of a central application. Java’s Remote Method Invocation allows for both. One stipulation to this method, is that passing a parameter requires that the parameter be a remote object as well.

1. *Fixed-sized and variable-sized messages*

In fixed-sized messages, a specific number of messages can be held by a buffer, as long as it has a specified size. This means that only a certain number of messages can be supported by the fixed buffer. This is also the disadvantage to fixed-sized buffers. The messages will not be supported if they are not a part of the fixed-sized buffer. Variable-sized messages do not have a specified size and therefore, it is unknown how many messages can be stored. This can be seen as the advantage to variable-sized messages, it can accept most sized messages. Also, variable-sized messages are using shared memory to communicate.

*Challenge Question Two:*

*Which of the following components of program state are shared across threads in a multithreaded process?*

*A. Register values*

*B. Heap memory*

*C. Global variables*

*D. Stack memory*

The answer to this question is both option B and option C. Register values and stack memory are private to each thread, therefore they would not be a good option for multithreaded processing. Global values and heap memory are capable of being shared amongst multithreaded processes.

*Challenge Question Three:*

*Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single-processor system? Explain.*

The answer to this question is no. A multithreaded solution could not achieve better performance using a multiprocessor because the operating system recognizes single processes and does not schedule different threads of a process on a different processor. The operating systems schedules kernel threads to run on physical processors (Silberschatz et al, 2009), which takes priority in the scheduler activation.

Reference:

Silberschatz, A., Galvin, P. B., & Gagne, G. (2009). *Operating System Concepts with Java* (8th ed.). John Wiley & Sons.