HW1

Written Exercise

1.9: Direct memory access is used for high-speed I/O devices in order to avoid increasing the CPU’s execution load.

(a) How does the CPU interface with the device to coordinate the transfer?

Commands and status requests are sent from the CPU to the device via some special registers, such as I/O ports or memory-mapped I/O registers. The device notices when these registers are written-to, and will take the appropriate action.

(b) How does the CPU know when the memory operations are complete?

The device could write to a special register or generate an interrupt. In either case, the processor notices what has happened.

(c) The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of the user programs? If so, describe what forms of interference are caused.

The CPU and the device will compete for cycles on the memory bus. The memory controller will try to fairly allocate bus cycles between the CPU and the device. So, any CPU program that would be capable of using all the memory bus cycles could run more slowly while the DMA is active.

2.7: What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?

There are two common models of interprocess communication:

(i) The message-passing model: The communicating processes exchange messages with one another to transfer information. A connection must be opened before communication can happen where a common mailbox allows the exchange of messages between the processes which can happen either directly or indirectly.

- Advantages:
    - It is useful for exchanging smaller amount of data
    - There are no conflicts to avoid
    - It is easier to implement
- Disadvantages:
    - Because of the implementation of the connection process described above, it is slower than its counterpart

(ii) The shared-memory model: Shared memory allows two or more processes to exchange information by reading and writing data in shared memory areas. The shared memory can be simultaneously accessed by multiple processes. Real-life examples would be the POSIX systems, as well as Windows operating systems.

- Advantages:

    - Allows maximum speed
    - It offers better convenience of communication
- Disdvantages:
    - It has security issues
    - The processes that use the shared memory model need to make sure that they are not writing to the same memory location.
    - Problems in the area of synchronization

2.10: What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?

(a) The microkernel approach modularizes the kernel and structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result would be a smaller kernel. Generally, microkernels provide minimal process and memory management in addition to a communication facility. Also, this approach uses message passing in order to allow communication between the client program and the various services that are also running in user space. As a result, the advantage of using the microkernel approach would be:
   (i) Extending the operating system is made easier.
   (ii) The operating system is easier to port from one hardware design to another.
   (iii) There are fewer changes needed to be made when modifying the kernel.
   (iv) The microkernel provides more reliability because of the simpler kernel design and functionality.
   (v) Since it uses message sharing to communicate and most services are running as a user rather than kernel processes, it also offers higher security.

(b) The microkernel must provide communication between the client program and the various services that are also running in user space. In this case, communication is provided through message passing. The client program and service never interact directly only by exchanging messages with the microkernel.

(c) One of the disadvantages of using the microkernel approach are the system-function overheads which affect the performance of microkernels. As described above, microkernels use message passing as a form to communicate and exchange information between the user process and the system services. This interprocess communication unfortunately comes with overheads because of the frequent use of the operating system’s messaging functions.

3.1: Describe the differences among short-term, medium-term, and long-term scheduling.

(a) Summary: Schedulers are special system software that handles process scheduling in various ways. Their main task is to select the job to decide which process to run first.

(b) Differences:

(i) Short Term Scheduler
It is also called a CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. The CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them. Short-term schedulers, aka dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

(ii) Medium Term Scheduler:
Medium-term scheduling is a part of swapping. It removes the process from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.
A running process may become suspended if it makes an I/O request. A suspended process cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

(iii) Long Term Scheduler:
It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.
The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.
On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of the long-term scheduler.

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

(a) Synchronous and asynchronous communication.

A benefit of synchronous communication is that it allows a rendezvous between the sender and receiver. A disadvantage of a blocking send is that a rendezvous may not be required and the message could be delivered asynchronously. As a result, message-passing systems often provide both forms of synchronization.

(b) Automatic and explicit buffering

Automatic buffering provides a queue with indefinite length, thus ensuring the sender will never have to block while waiting to copy a message. There are no specifications on how automatic buffering will be provided; one scheme may reserve sufficiently large memory where much of the memory is wasted. Explicit buffering specifies how large the buffer is. In this situation, the sender may be blocked while waiting for available space in the queue. However, it is less likely that memory will be wasted with explicit buffering.

(c) Send by copy and send by reference

Send by copy does not allow the receiver to alter the state of the parameter; send by reference does allow it. A benefit of send by reference is that it allows the programmer to write a distributed version of a centralized application. Java's RMI provides both; however, passing a parameter by reference requires declaring the parameter as a remote object as well.

(d) Fixed-sized and variable-sized messages

The implications of this are mostly related to buffering issues; with fixed-size messages a bugger with a specific size can hold a known number of messages. The number of variable-sized messages that can be held by such a buffer is unknown. Consider how Windows 2000 handles this situation: with fixed-sized messages (< 256 bytes), the messages are copied from the address space of the sender to the address space of the receiving process. Larger messages (i.e. variable-seized messages) use shared memory to pass the message.