Homework № 1

Steven Seppala 21 September 2014

Question 1.5

There are two modes of operation.

1) Kernel Mode
2) User Mode

Kernel mode is the privileged mode. If the mode bit is in the low state (0), then the system is in kernel mode. By using this and also making some instructions privileged, we can protect the system from users who either do not know how to use some commands that can be dangerous, and we can protect other users from those same users who might be malicious or just not know what they are doing. These instructions are operations such as IO interfacing, task scheduling, and other such operations.

User mode is when the mode bit is high (1). When we are in user mode, we can not perform privileged operations. If we try to, the operating system raises exceptions and tells the user that they are not allowed to execute that instruction.

Problem 1.6

1. Privileged.
2. Not - Privileged.
3. Privileged.
4. Not - Privileged.
5. Privileged.
6. Privileged.
7. Not - Privileged.
8. Privileged.

Problem 1.20

1. Direct memory access command blocks are written to main memory which contains pointers to the source and destination of the transfer and a number of bytes that are to be transferred. The CPU writes the address of the command block to the DMA controller. It can start a DMA operation by writing values into special function registers. The device then starts the operation once it comes from the CPU. When it is finished it interrupts the CPU to tell it that the operation is finished. The OS uses device drivers to watch the hardware controllers that do this, the device drivers have special registers and buffers so that the CPU isn't tied down with these operations.

2. The device controller sends an interrupt to the CPU. Both the device and the CPU can access the memory simultaneously. The memory controller provides access to the memory bus in a fair manner between the two.

   The CPU starts the transfer by creating the interface with the correct arguments to be transferred. The controller reads the block and into the buffer and verifies the check-sum, then begins to copy the block into the memory given to it by the DMA. Then it increments the DMA address and decrements the DMA count. Once the count is zero the process is finished and an interrupt is generated.

3. The CPU isn't allowed to execute other programs while the DMA is transferring the data. It can still access data in the caches where there is no interference with user programs; such as primary and secondary caches. When the DMA sends the interrupts it can cause a user process to be suspended.

Problem 2.6

First a fork() command must be called, followed by exec(). The fork() command creates a copy of the process which called it, including it's file pointers (std in, std out, and std err). Then exec() loads the program that is called into memory.

Problem 2.21

It provides minimal process and memory management with communication facility. The communication is done indirectly through message passing. All new processes are added to the user space and do not require modifications of the kernel. It provides more security and reliability since the majority of services running are the user's instead of the kernel's.

However, the main disadvantage of these is performance reduction due to increasted system overhead.

Problem 3.2

The code produces eight (8) process in total, counting the parent as well.

Problem 3.14

ppid = 2600
child pid = 2603

Line A : "child : pid = 0" .
Line B : "child : pid1 = 2603".
Line C : "parent : pid 2603".
Line D : "parent : pid1 = 2600".

Problem 3.18

1. Synchronous communication allows rendezvous between the sender and receiver. A benefit is that neither process has to block its execution, which gives better performance.

   A drawback is that blocking the send could lead to uncertainty when it will be delivered and it could be delivered when the receiver no longer has interest.

2. Automatic buffering provides a que with unknown length. This ensures the sender will never have to block while waiting to copy a message. There are no specifications on how this is implemented though, one specification may waste enormous amounts of memory.

   Explicit buffering dictates how long the buffer is to be. The sender may be blocked while waiting for space in the que; but it is less likely that memory will be wasted while explicitly buffering.

3. Send by copy is better for network generalization and synchronization. It doesn't allow the receiver to alter the sate of the parameter but send by reference does. Send by reference is more efficient for larger data but hard to code for because the programmer must write shared memory implications.

4. The implications are mostly related to buffering fixed size messages. Fixed size messages are easier to implement at the kernel level but require more programing.
   Variable sized messages are more complex for the kernel but easier to program for. The number of variable sized messages that can be held by a buffer is dependant on how long the messages accumulate to be, and thus is unknown until it is known.

Problem 4.7

If a web server for instance ran only a single thread, then it could only service one client at a time; making the time between clients receiving data possibly enormous. It is generally better to run this as a process that contains numerous threads.

Problem 4.17

Line C : "CHILD : value = 0"
Line P : "PARENT: value = 5"