Operating System HW1 110590450 歐佳昀

- Chap.1

1.16: *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?

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

(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.

Ans:

(a) CPU will set up the DMA registers, which contain pointers to the source and destination of the transfer, along with a counter indicating the number of bytes to be transferred. Then the DMA controller proceeds to place addresses on the bus to perform transfers, while the CPU is available to accomplish other work.

(b) When the entire transfer is finished, the DMA controller interrupts the CPU.

(c) Both the CPU and the DMA controller are bus masters. so when they want to to access the memory at the same time, it can lead to problems. therefore, if the DMA controller seizes the memory bus, we should stop the CPU momentarily to prevented from accessing main memory .

However, if the CPU is still Permitted to access data in its primary and secondary caches, and both the CPU and the DMA controller update the same memory locations, it may cause a coherency issue.

- Chap. 2

2.15: What are the two models of *interprocess communication*? What are the strengths and weakness of the two approaches?

2.19: 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?

Ans:

2.15 : Message-passing model and the Shared-memory model.

Message passing is suitable for exchanging smaller amounts of data because conflicts need not be avoided. It is also easier to implement than shared memory for inter-computer communication. Shared memory allows for maximum speed and convenience of communication because it operates at memory transfer speeds within a computer.

However, this method compromises on the protection and synchronization between processes sharing memory.Need to ensure that they are not writing to the same location simultaneously.

2.19 :

(1) It can add a new service does not require modifying the kernel, and a simpler kernel design. Also, it provides more security and reliability, since most services are running as user rather than kernel processes.

(2) using interprocess communication mechanisms such as messaging. These messages are conveyed by the operating system

(3) microkernels can influence from performance decreases due to increased system function overhead. and the frequent use of the operating system’s messaging functions , in order to enable the user process and the system service interation

- Chap. 3

3.12: Describe the actions taken by a kernel to context-switch between processes.

3.18: Give an example of a situation in which ordinary pipes are more suitable than named pipes and an example of a situation in which named pipes are more suitable than ordinary pipes.

Ans :

3.12 : Context switching between kernel threads should save the state of the currently running process ,restore the state of the process scheduled which run next, and save including all the CPU registers values in addition to memory allocation. should do many architecture-specific operation,

3.18 : example:

Suitable for ordinary pipe : could be used where the producer writes the file to the pipe and consumer reads the files and counts the number of characters in the file . the communication needs to happen only between two specified process, known beforehand. Named pipes in such a scenario would involve too much of an overhead.

Suitable for named pipe : writing a message to a log and then a server reads the message from the named pipe and writes onto the log file. listen to requests from other processes. If the calling processes are aware of the name, they can send requests to this.