Since contemporary hardware has the capability to control many facets of I/O operations autonomously and without direct involvement of the main system processor, significant performance benefits may be attained when input/output devices are operated in concurrent fashion. Concurrent I/O operation is accomplished by initiating a number of independent I/O operations before others in progress are completed. On the other hand, concurrent execution of independent tasks requires careful scheduling of many different code sequences and of time-critical events sensed or caused by input/output devices.

The goal of this chapter is to explore the principles of the input/output subsystem, to study the elements of I/O programming, and to introduce the increasing degrees of concurrent execution of I/O operations that the hardware supports. The detailed treatment of management of multiple devices via polling and hierarchical interrupts provides a foundation for the implementation of a kernel, presented in Chapters 10 and 11. Readers familiar with the intricacies of interrupt structure and I/O programming, or those only marginally interested in implementation details of the multitasking kernel presented in subsequent chapters, may omit this chapter without loss of continuity.

9.1 THE INPUT/OUTPUT PROBLEM

In this section we focus on the hardware aspects of the input/output subsystem and stress its unique attributes, which both necessitate specialized programming and make concurrent operations possible.

In general, most I/O devices share two important characteristics that largely define the input/output problem:

1. Asynchronous operation
2. The speed gap

The following sections treat each of these characteristics in detail.

9.1.1 Asynchronous Operation

During program execution, the processor engages in numerous transactions with the main memory and I/O devices in order to fetch instructions for execution and to access data operands. For example, READ operations with both memory and devices are typically initiated by the processor, which, some time later, obtains the required items. However, the timing of memory and that of I/O transactions are fundamentally different.

Except possibly when errors occur, memory is always ready to furnish a datum whose address is designated by the processor. It does so within a nearly constant time, the *memory-access time*, which generally lasts at most a few cycles of the processor clock. In that sense, memory may be regarded as operating synchronously with the processor. Another important observation is that the processor waits until the desired item from memory arrives, as the processor has nothing to do until an instruction or datum is fetched.

Since contemporary hardware has the capability to control many facets of I/O operations autonomously and without direct involvement of the main system processor, significant performance benefits may be attained when input/output devices are operated in concurrent fashion. Concurrent I/O operation is accomplished by initiating a number of independent I/O operations before others in progress are completed. On the other hand, concurrent execution of independent tasks requires careful scheduling of many different code sequences and of time-critical events sensed or caused by input/output devices.

The goal of this chapter is to explore the principles of the input/output subsystem, to study the elements of I/O programming, and to introduce the increasing degrees of concurrent execution of I/O operations that the hardware supports. The detailed treatment of management of multiple devices via polling and hierarchical interrupts provides a foundation for the implementation of a kernel, presented in Chapters 10 and 11. Readers familiar with the intricacies of interrupt structure and I/O programming, or those only marginally interested in implementation details of the multitasking kernel presented in subsequent chapters, may omit this chapter without loss of continuity.

9.1 THE INPUT/OUTPUT PROBLEM

In this section we focus on the hardware aspects of the input/output subsystem and stress its unique attributes, which both necessitate specialized programming and make concurrent operations possible.

In general, most I/O devices share two important characteristics that largely define the input/output problem:

1. Asynchronous operation
2. The speed gap

The following sections treat each of these characteristics in detail.

9.1.1 Asynchronous Operation

During program execution, the processor engages in numerous transactions with the main memory and I/O devices in order to fetch instructions for execution and to access data operands. For example, READ operations with both memory and devices are typically initiated by the processor, which, some time later, obtains the required items. However, the timing of memory and that of I/O transactions are fundamentally different.

Except possibly when errors occur, memory is always ready to furnish a datum whose address is designated by the processor. It does so within a nearly constant time, the *memory-access time*, which generally lasts at most a few cycles of the processor clock. In that sense, memory may be regarded as operating synchronously with the processor. Another important observation is that the processor waits until the desired item from memory arrives, as the processor has nothing to do until an instruction or datum is fetched.

Since contemporary hardware has the capability to control many facets of I/O operations autonomously and without direct involvement of the main system processor, significant performance benefits may be attained when input/output devices are operated in concurrent fashion. Concurrent I/O operation is accomplished by initiating a number of independent I/O operations before others in progress are completed. On the other hand, concurrent execution of independent tasks requires careful scheduling of many different code sequences and of time-critical events sensed or caused by input/output devices.

The goal of this chapter is to explore the principles of the input/output subsystem, to study the elements of I/O programming, and to introduce the increasing degrees of concurrent execution of I/O operations that the hardware supports. The detailed treatment of management of multiple devices via polling and hierarchical interrupts provides a foundation for the implementation of a kernel, presented in Chapters 10 and 11. Readers familiar with the intricacies of interrupt structure and I/O programming, or those only marginally interested in implementation details of the multitasking kernel presented in subsequent chapters, may omit this chapter without loss of continuity.

9.1 THE INPUT/OUTPUT PROBLEM

In this section we focus on the hardware aspects of the input/output subsystem and stress its unique attributes, which both necessitate specialized programming and make concurrent operations possible.

In general, most I/O devices share two important characteristics that largely define the input/output problem:

1. Asynchronous operation
2. The speed gap

The following sections treat each of these characteristics in detail.

9.1.1 Asynchronous Operation

During program execution, the processor engages in numerous transactions with the main memory and I/O devices in order to fetch instructions for execution and to access data operands. For example, READ operations with both memory and devices are typically initiated by the processor, which, some time later, obtains the required items. However, the timing of memory and that of I/O transactions are fundamentally different.

Except possibly when errors occur, memory is always ready to furnish a datum whose address is designated by the processor. It does so within a nearly constant time, the *memory-access time*, which generally lasts at most a few cycles of the processor clock. In that sense, memory may be regarded as operating synchronously with the processor. Another important observation is that the processor waits until the desired item from memory arrives, as the processor has nothing to do until an instruction or datum is fetched.