6 Input and Output

At this stage in the development of our paper operating system we have established an environment in which processes can exist and in which memory can be allocated to hold the associated programs and data. We now turn our attention to the means by which processes communicate with the outside world: that is, to mechanisms for the input and output of information.

Traditionally I/O is regarded as one of the more sordid areas of operating system design in that it is a field in which generalisation is difficult and ad hoc methods abound. The reason for this is the wide variety of peripheral devices employed; a particular configuration may include devices which differ greatly in their characteristics and mode of operation. Specifically, devices may differ on one or more of the following counts.

(1) **Speed**

There may be a difference of several orders of magnitude between the data transfer rates of various devices. A magnetic disc, for instance, may be able to transfer $10^6$ characters per second, compared with a terminal keyboard speed of only a few characters per second (depending on the typist!).

(2) **Unit of transfer**

Data may be transferred in units of characters, words, bytes, blocks or records, according to the peripheral used.

(3) **Data representation**

An item of data may be encoded in different ways on different I/O media. Even on a single medium, such as magnetic tape, several different codes may be employed.

(4) **Permissible operations**

Devices differ in the kind of operation they can perform. One example is
the obvious distinction between input and output; another is the ability to rewind magnetic tape but not line printer paper.

(5) Error conditions

Failure to complete a data transfer may have various causes such as a parity error, a printer wreck, or a checksum error, depending on the peripheral being used.

Clearly the diversity exemplified above is difficult to handle in a uniform manner. However we shall attempt in this chapter to construct a framework for an I/O system in which device-dependent characteristics are isolated as far as possible and in which some degree of uniformity is achieved. We start by considering some design objectives and their implications.

6.1 Design objectives and implications

(1) Character code independence

It is obviously undesirable that in order to write programs a user should require detailed knowledge of the character codes used by various peripherals. The I/O system must take responsibility for recognising different character codes and for presenting data to user programs in a standard form.

(2) Device independence

There are two aspects to device independence. First, a program should be independent of the particular device of a given type which it happens to be allocated. For example, it should not matter on which magnetic tape deck a particular tape is mounted, or which line printer is used for a program's output. Device independence of this kind ensures that a program does not fail simply because a particular device is broken or allocated elsewhere. It gives the operating system the freedom to allocate a device of the appropriate type according to overall availability at the time.

Second and more demanding, it is desirable that programs should as far as possible be independent of the device type used for their I/O. Clearly device independence of this nature cannot be carried as far as sending output to an input device; what we have in mind is that only minimal changes to a job should be required for it to receive its data from disc rather than magnetic tape.