WHAT'S AN OPERATING SYSTEM ?

The first task is to get some idea of what we're talking about. The culmination of the quest for efficiency sketched in the preceding chapter was the batch operating system, but perhaps the best known representative of this stage of development – because many more people actually used it – was the *time-sharing* system. In this chapter, therefore, we say a little about such a system in order to give a hint of where we are going.

WHAT DO WE NEED ?

Here's a description of a fragment of a terminal session on a fairly ordinary time-sharing system, starting from the moment when someone starts trying to log in to the system to do some work. On the left, the events; on the right, the system's response, and, in *italic*, a hint of the system components concerned.

A key is pressed on a vacant terminal.	An electrical signal representing a character sent from terminal to computer must be handled immediately; it is typically stored in a buffer until the end of the instruction. *Interrupt handlers; the terminal device driver.*
Some special key (commonly RETURN) marks the end of the instruction.	The accumulated line of characters must be sent to a programme which can interpret it. *The command interpreter*.
A LOGIN instruction is found.	A programme which deals with starting up a new session must be started and pointed at the terminal. *The login programme.*
The login programme checks that the person is permitted to use the computer system.	The person's characteristics are compared with filed information. *The user information file.*
The process is repeated to read another instruction.	The corresponding programme is loaded for execution. *Disc manager, memory manager.*

And so on. All these things (and plenty more) have to be done quite irrespective of the nature of the programme which is to be executed; they are all the responsibility of the operating system.

There are rather a lot of housekeeping tasks which you don't want to be bothered with when you're using a computer. Some of them are things you know about : running programmes, copying disc files, and handling input and output between the computer and the screen are all jobs which you know you want done, but which you don't want to have to do for yourself. Others, which you might or might not know about, are (at least comparatively) invisible : providing areas of memory for you to use, knowing where the disc files are, running many programmes at once on the same processor. Then there are administrative tasks which must be carried out : keeping accounts, recording statistics. All these have to be done somehow on some computers simply so that the machines can be used to best advantage; and they are all the job of the operating system.

So the operating system is a very varied collection of computer procedures which we need to make the computer hardware usable. They are the procedures which lie underneath all the programmes we actually want to run on the computer, providing support services for everything else that goes on.

ANOTHER VIEW.

Or is it ? What are computers for, anyway ? They're for helping people to get work done. (Well, that's the theory …) But raw computers are not much use to anyone – or,

perhaps better, no use to almost everyone. Most people need some help to get programmes running in computers : we need something between us and the raw machine which will accept our instructions in some reasonably comprehensible language, and put them into action, at the same time making sure that all the resources needed are provided as and when required, and that any errors or difficulties are reported back to us, again in comprehensible terms. That's the operating system too.

But this time it's the procedure that provides service to someone who wants to use the machine; it's in charge of the programmes we run, not their servant.

FUNCTIONAL SYSTEMS.

That was essentially a (very sketchy) definition of an operating system in terms of the services it must provide. We also want to say something about *how* the services should be provided. There are all sorts of things we can say about that, too, but the most important is the very basic requirement that programmes run under the operating system's control should work properly. That might sound trivial; but plenty of systems in widespread use nowadays (including the one on which this text was first typed) don't match up to it.

We can approach a definition by starting with the idea of a *functional programme* (which has nothing whatever to do with functional programming). A functional programme corresponds to the mathematical notion of a function, in that its behaviour depends only on its input – so a programme which computes different results by using whatever happens to turn up as the value of an uninitialised variable isn't functional. Functional programmes are, almost always, what we want.

A *functional system* is essentially one which guarantees that a functional programme will run properly under all circumstances. The operating system, in administering the execution of programmes, has lots of opportunities to get things wrong. It can allow independent programmes to interfere with each other, or lose track of what's going on because it receives too many interrupts from outside too often, or otherwise mess up a perfectly good programme. To require that the system be functional is a shorthand way of saying that it won't do any of these nasty things to your programmes.

SOMETHING SLIGHTLY DIFFERENT.

We've assumed that an operating system's primary job is to deliver services to people, but that isn't always so. Many computers spend their time on jobs which don't directly involve people – such as controlling machinery. They need operating systems too, but their requirements are different (though there is, of course, a lot of overlap), and we don't discuss them in the 07.340 course.

QUESTIONS.

Think of some other things that you want to be done for you as you use a computer.

What do YOU think an operating system is ?

Criticise some other views : to control users; to manage resources; anything else plausible that comes to mind.
