HOW TO USE A TERMINAL

The archetypal computer terminal was a Teletype machine, which is not much different from a traditional mechanical typewriter with electrical additions which can detect when keys are pressed and transmit corresponding encoded signals along a wire, and also receive electrical signals from the wire and cause these to drive solenoids which operate the corresponding keys. And that's all. You could do anything you liked with it, provided that it could be done with the standard characters and a few control operations such as backspace, carriage return, and line feed. You couldn't wind the paper back, and you couldn't untype characters, so each typed line could only get blacker and blacker until you moved to the next line below, at which point the line you had just left was out of reach for ever.

Under these constraints, the scope for clever user interface design was rather limited, but it wasn't zero. It was big enough for the development of three styles of operation which are still around. These are sometimes called *encoding* and *selection*, used mainly for giving specific instructions, and *prompting*, used to provide arbitrary information. To get something done using encoding, you have to know how to do it – so if you want to edit a file called X, you must know the correct instruction (say, **edit X**). Getting something done using selection is a cooperative effort between you and the system : the system presents you with a list of possibilities, from which you select one. Prompting is also cooperative – the system asks a question, to which you give an answer.

We mention these facts to make an important point : encoding, selection, and prompting are *still* the only ways we have to use terminals. We have cleverer ways of doing them now, but we still do them. The ways in which we can use terminals and the physical forms of the terminals themselves are not completely independent, but they are fairly close. Here are some examples showing how the different techniques fit in.

TEXT – THE COMMAND LINE.

The command line interface is the traditional way to give instructions to an operating system. You give the computer an instruction by typing a string of text. It is simple, familiar, and cheap. It demands simple typing skills for efficient use, but most people seem to be able to cope reasonably well. It is simple encoding : you have to know the right way to form the instructions (say, **edit X**), and you simply enter them at the keyboard. If the instruction is faulty in any way, you should receive an error message, but just what that's like depends on the system. The big advantage of this method is that it will work immediately for any instruction, provided that you get it right, so it can be very effective for anyone experienced in using the system; also, there is no overhead for constructing displays, formatting, etc. The disadvantage is that if you forget the instruction there's nothing intrinsically helpful about the interface – though in practice there is usually some sort of help system available.

MENUS.

In a menu interface, the possible choices are presented (possible, if inconvenient, with a teletype, but practicable with a screen) and you pick one. The simple way to select your choice is to enter its number or other identifying symbol at the keyboard; that's possible with any terminal. Once you can move a cursor around a screen with arrow keys from the keyboard, you can select by placing the cursor at the appropriate position and (typically) pressing <return>; and when you have a mouse which can move a pointer, everything becomes easy.

But all these are essentially the same method. They all depend on cooperation between system and person, in which the system presents a set of possibilities, from which one can then be chosen. This method works well if the number of possibilities is comparatively small (so that the menu will fit on a screen), and automatically includes a reminder of the set of possibilities. On the other hand, it takes time for the system to draw the menu and for you to make the selection; if you construct the system in the obvious way it can waste a lot of time for someone who knows which choices to make without looking at the menus.

FORMS.

Menus work well if the answers are known in advance, but they are not very useful for gathering general information. This is where prompting is of value : the system displays a question, and you enter the answer from the keyboard. There is still no more satisfactory way to enter an arbitrary name (personal, file, or whatever), or anything else for which it's impossible to predefine a sensible list of possibilities. A form is just a collection of prompts, to which you respond in the obvious way.

Forms interfaces can be very effective, and they turn up today not only as "pure" forms, but as the dialogue boxes well known in WIMP systems.

WIMP.

WIMP interfaces are the conventional (Macintosh, Windows, X-windows, etc.) graphical interfaces which depend on a mouse and a graphics screen. They are included here for completeness, but are sufficiently demanding to justify special treatment; we shall have much more to say about them in the next chapter.

QUESTIONS.

How could you speed up menu selection for someone who knows the answers ? Consider the separate virtues of selection by mouse and selection by keyboard entry.
