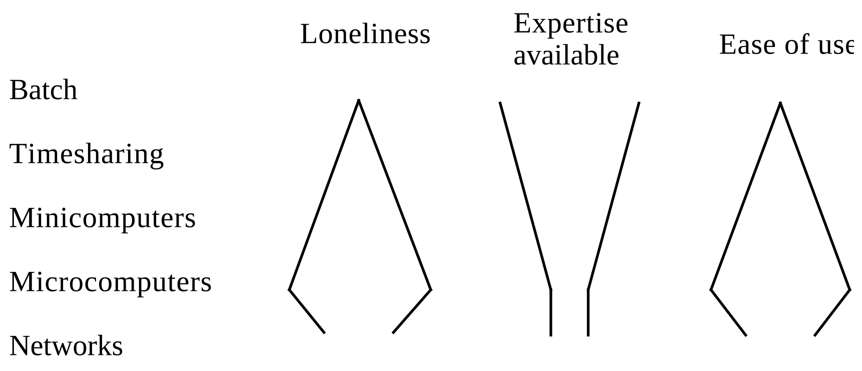
USABILITY : GETTING THE MOST OUT OF THE PEOPLE.

In an earlier chapter entitled *EFFICIENCY : GETTING THE MOST OUT OF THE MACHINE*, we introduced the pursuit of efficiency, and asserted that it had governed the first half of the development of operating systems. Batch systems are the culmination of that process; over the years, they evolved into impressively effective and efficient ways of running computers, and they are still used for many purposes. But their reign over the whole computing world was of rather short duration. As we saw earlier, changing costs saw the cost of computing – and, eventually, most other – people exceed the cost of computing hardware, and the focus for decreasing costs had to move to increasing people's efficiency instead of machines'. This is the principle which has governed developments during the second half of operating system history.

This new sort of development is no longer a simple overlapping of operations, with automation of tasks which people could only manage too slowly. This worked wonders for the batch systems, but the result was systems which did very much the same sort of work as those early slow systems from which they grew, and were expected to carry on doing much the same. In the new systems, computers begin to take on different sorts of work, which would earlier have been regarded as a waste of their time. To emphasise the significance of the change, consider that in the 1970s postgraduate students at Auckland university were not permitted to use the university's computer for what we would now call word processing; computing time was much too expensive to waste on mere clerical work. (We would now call it very primitive word processing, but the formatting programmes which presented typewritten text nicely – notably NROFF on Unix systems – looked pretty good at the time.)

Beneath this change, the principle remains the same : we are still trying to identify the least efficient part of the operation, and do something about it. The differences are, first, that the least efficient part has moved out of the computer, and might be the secretary's (or postgraduate student's, who are very cheap indeed) retyping reports and theses, and, second, that we can now use the comparatively cheap computer to try to do something about it.

The progression, particularly as it affected the people who used the system, is symbolically depicted in the diagram below; we discuss the more technical aspects of the changes later.



Batch systems ran in computer centres, which to some extent also became social centres. You had to carry your cards there, and collect your printed output, so you'd get to know other people using the system, and the computer centre staff. If you wanted help, lots was available. When interactive services became available through timesharing systems, interaction between people declined sharply; now you could use a terminal in your own department, or even your own room, so you didn't meet other people or experts. You could instead talk to people through the terminals, and the beginnings of electronic mail grew up, but mainly between people at a single site, all sharing the same processor. Systems did become rather easier to use, but mainly as a result of removing difficulties than by actively seeking to help people.

With minicomputers, contact with the rest of the world diminished even further. If you didn't have your own expert (or, more commonly, amateur expert, which was sometimes good enough), you didn't do it. Electronic mail stopped. The trend towards more easily used software continued, though still with little concession to the non-technical. That came with microcomputers, when terms such as "user-friendly" became common, and no more honest than they are today. At least, though, there was an attempt to make interfaces comprehensible to real people who hadn't spent years learning about programming. Nothing particularly new has happened since, though the machines themselves have become smaller, more powerful, and more various. We have heard of lap-top computers, palm-top computers, personal digital assistants, and other silly names; some of them have operating systems which deviate significantly from the traditional pattern, but from the usability point of view the same trends have continued.

Networks are the most recent significant development. These have brought back electronic mail (quite some time ago now, though the internet in its present form is less than ten years old – in 1988, it wasn't straightforward to exchange electronic mail between England and New Zealand), and many new and more elaborate variants of that and similar communications techniques with the World-Wide Web only the latest of a series of novel techniques. When these sort themselves out, we shall have a range of methods for communicating data of many kinds between machines, and ways of running many machines together as integrated systems. What we don't know yet is how we can best use the systems.

As with all other developments in computing, there are many prophets ready to tell you that "the information superhighway" (anything that needs a name like that really must be pretty rubbishy !) will solve all our problems. We don't believe it – it's never been true before. We confidently expect that the fully developed network systems will solve some of our problems, and create some new ones. Some of us will end up a little better off, perhaps financially, perhaps in access to information or other people, perhaps in other ways. There will almost certainly be new opportunities for crime, which the criminals will find first, then the law-enforcement agencies, then – a long time later – the legal profession. Eventually, it might even seep through to politicians, but don't bank on it.

In short, international networks are tools, and we'll use them as just as wisely and well as we use other tools.

INTERACTIVE SYSTEMS.

The first dent in the batch systems' monopoly was the appearance of *interactive* systems, in which jobs punched onto cards were replaced by jobs submitted from terminals. While terminals themselves were not new, they had previously been used either as operators' consoles or for programmes' input and output in the same way as other devices. The new feature was the use of terminals for both input and output and direct communication with the operating system. From the point of view of efficient use of the central processor, this is a step backwards; it is never possible to achieve acceptable response to a terminal while at the same time using most of the processor time profitably. The aim of an interactive system is to make more efficient use of the *people* who are working with the computer. Whether or not it is as successful as is sometimes claimed is a matter of debate, but the significant feature is in the shift of viewpoint.

MICROCOMPUTERS.

The development of minicomputers around 1970 to 1980 made little difference to operating systems practice. Those which had anything corresponding to an operating system followed very much in the footsteps of the large machine systems, typically demonstrating their authors' inability to learn from their predecessors' mistakes. The only major difference was the direct assumption that terminals would be the major form of communication with the computer.

The advent of microcomputers from about 1980 onwards began in the same way, and once again the same mistakes were evident. Nostalgia reigned. The system software was largely at the level of the old monitor systems – and in many ways still is – but the pattern changed in two ways.

First, the microcomputers became so cheap, and the available software so good, that they could provide affordable services to almost anyone who wanted them, and they began to rival the large machines. It became feasible to think of satisfying an organisation's computing requirements using a lot of microcomputers, perhaps linked together by the rapidly developing networking techniques, or simply by people walking about carrying floppy discs. In itself, that didn't make much difference to the system software, but it made it worth while investing a lot of development effort in the microcomputer area.

The other change was in the way the system software interacted with the person using the machine. Because the microcomputers were so cheap and so widespread, it was no longer sensible to assume that anyone using the machine would be a trained computist, and would understand messages couched in technical computer gibberish. (It never had been very sensible, but was universally done anyway.) If the new breed of "naive user" were to be dissuaded from throwing away the computer in well justified disgust – or, at a more petty but frequently more persuasive level, suing the manufacturer for something or other – communication between system and computer would have to be conducted in something a good deal closer to a human language. The idea that computers should be – or even could be – easy to use was new, but sowed the seeds of developments that have led to systems like the Macintosh, and other adventurous sorts of interface.

We are still concerned with "getting the most out of the people" – but the people have changed. The interactive systems were built to optimise the performance of professional computists; the newer systems are at least designed with the intention of increasing the computing performance of the population at large.

ANOTHER INTERFACE.

But wait a minute ! The professionals are still here, and they still haven't regained some of the ground they lost when operating systems arrived. One of the useful features of a monitor system which disappeared when operating systems started to become paranoid about their possessions was easy access to the system facilities. While it is true that the restrictions were imposed because some of the undisciplined access to the system could be dangerous, the result of the changes was to prevent pretty well all access to the system, dangerous or not. In effect, a programme became something very close to a sealed box, which could communicate with the outside world only through the recognised input and output channels which connected it with devices.

This wasn't very satisfactory, and led to silly constraints. Frequently it was impossible to write a programme which would perform actions which you could perform from the keyboard as a matter of course – so, for example, you might be able to check the type of a file by a keyboard instruction, but be quite unable to encode the same operation into a programme. As the programmes are intended to help you to use the computer, that isn't at all constructive; and as you could often write a "command file" – essentially a programme of system instructions executed by the operating system – which will perform the instructions which you can't do from the ordinary programme, the restrictions are obviously silly.

At that level, the restrictions are irksome, but survivable. It certainly led people (including at least one of us) to go to considerable lengths to split up operations between programmes and command files, writing several smaller programmes to be executed – from the command file – in between the system operations. With later developments, most notably with the introduction of graphics terminals, it was essential in the interests of consistency that programmers should be able to use large number of system facilities easily but securely, and it became important to find an acceptable way to satisfy this requirement.

One way out of this difficulty is to expand the supervisor call mechanism. By this means, access to system operations can be provided in a safe and controlled way. Another is to provide what amounts to a subroutine library, with the various system functions reached as procedure calls. This leads to the idea of the *application programme interface* (usually API), and most systems nowadays provide extensive interfaces of one sort or another through which programmers can gain access to useful provided functions. APIs will turn up again from time to time in our discussions, but they weren't always there.