WORKING NOTE D5

Alan Creak
7 June 1986

A TWIN PROCESSOR SYSTEM

This note was left as unfinished business in 1986, and forgotten. I have now (2003) reconstructed it from an essentially empty old MacWrite file, adding a reverse-engineered version of what I think I was going to write assisted by a very scrappy note, and undoubtedly to some degree by hindsight. There is also a certain amount of unambiguously 2003 commentary, presented in this typeface. This note is likely to be of historical interest only, if at all, but it's nice to have the series complete.

THE CONTEXT.

On 7th June 1986 I was still trying to find out about rehabilitation computing. My main source of information was a couple of reports about circumstances in the USA, given to me by Neil Scott when I visited him in January 1986, and a survey on how disabled people in New Zealand were using computers. I'd visited the Wilson Home, and had picked up bits from various journals to do with robotics and artificial intelligence.

1. Reliability, backup.

Microprocessors are pretty reliable devices, but they do sometimes break down. In such instances, a system with two processors could be in a good position to survive. This would be particularly valuable in cases where the computer system was involved in work which it would be dangerous to stop – life support systems, wheelchair control, etc..

An obvious, but important, condition must be satisfied if the advantages of redundancy are to be realised : whatever else the two processors might be used for, it must be possible to run vital operations with just one if necessary.

A second condition determines when the processors can also be run separately, perhaps with one engaged on work quite unrelated to essential matters. In such cases, it must be possible automatically to detect the breakdown of either processor, to suspend non-essential work, and to pick up the essential activities with the remaining processor. Automatic detection is necessary, as the changeover must be effected sufficiently quickly that control systems should carry on effectively uninterrupted, even if they have been transferred to the other processor.

2. Terminal emulation.

A serious problem in the early development of rehabilitation computing was the practical impossibility of modifying existing software to work with special interface devices. Software for portable computers was (almost) invariably written to operate with specific devices – typically (obviously) keyboard and screen – with no simple provision for changing them.

That does not mean that change was impossible, but subterfuge and subversion was essential. The common practice of "memory mapping" the device interfaces made it fairly easy to intercept the communications, but it was usually difficult to find a place for the software which managed the interception. A common arrangement for a microcomputer programme was to reserve the top and bottom ends of memory for static elements (code, Basic interpreter, device driver software), and to satisfy other requirements – programme stack, symbol table, etc., as required. Whether or not there was an area which could safely be used for additional code, and, if so, where it was were questions which were not easy to answer.

Another processor, preferably with some memory of its own, would be of considerable assistance. So far as I know, nobody ever tried it, but there was a great deal of activity by many enthusiastic people, and comparatively little disciplined publication, so it might well have happened. There is more than one way to approach a solution, but the potential for improvement is certainly there.

3. One VM, one actual programme.

This suggestion was connected with my proposal[1] for using a virtual machine (VM) to provide a standard environment for rehabilitation systems, but I can no longer recall the details. My best guess is that this is an extension of the terminal emulation idea, with the major aim of getting the addition software – in this case, the virtual machine – out of the way of the executing programme. Given that much of the software available was constructed with the assumption that the whole of memory was theirs to control, and the common restriction of a microprocessor's memory to 64 kbytes, that wouldn't be a bad trick. I can't remember how I proposed to make it work.

REFERENCES

1 : G.A. Creak : *Proposed project : a virtual machine as a standard* (unpublished Working Note D2, 3 May 1986).