Computer Science 340

Operating Systems

SECOND TEST, 1991 : SKETCH ANSWERS

Remarks in italic type were added after marking the test.

QUESTION 1.

(a) Different languages and other systems want to structure memory in different ways; such a memory structure is called a memory model. For example, Algol-like languages rely on a stack model of memory.

Software support for "foreign" memory models is possible, but exceedingly expensive, because it requires action at every attempt to use memory. Software memory support is therefore usually restricted to services directly connected with the native addressing model of the computer hardware.

> *A memory model is rather more than a handy way to describe the operation of part of a programme. It amounts to an assumption about how the computer hardware works, so it must apply to everything that goes on. ALL of Algol (etc.) is based on the model of stacks-of-activation-records.*
>
> *A memory model is not a way of mapping anything onto anything. See above. If your memory model is different from the memory model implemented in the hardware,* ***then*** *you need a way of mapping, etc.*
>
> *A conventional virtual memory system isn't satisfactory as an example, because its purpose is precisely to implement a FLAT memory model. Processes in a conventional system using virtual memory still "see" a flat memory. The virtual memory system implements a lot of flat memories in a single flat memory.*

(b) The Linda tuple memory supports the storage and retrieval of ordered sets of objects (tuples). It must be possible to insert tuples, delete them, and search for a tuple matching a presented pattern.

Process A sends a message to process B by inserting a recognisable tuple into the memory - for example, { Tuple-for-B, from-A, data }. B can then receive the message by searching for a tuple which matches { Tuple-for-B, from-A, ? }, or perhaps { Tuple-for-B, ?, ? } if it wishes to receive from any of several different processes. B can then either leave the tuple in the Linda memory, or delete it.

> *I expected at least a strong indication that you knew (and I preferred an explicit description of) how the tuple matching operation was conducted; it's very important for the operation of Linda. People writing vaguely about labels which had to be the same lost a mark unless there was some other evidence that they knew what they were talking about.*

(c) The proposed method would provide all the necessary memory functions, but unless special measures are taken is totally lacking in security, and therefore quite unsatisfactory. To provide a secure system, it is far better to make a fresh start rather than to try to patch up a scheme with fundamental dangerous features built in. It is therefore more appropriate to use a completely separate memory space for the tuples, providing access through system calls in the same way as for other resources which need protection.

*Page **maps** should have been called page **tables**. I think only one person who answered this part was confused by the mistake - but that isn't unbiased evidence. As processes don't have page maps but do have page tables, there was strong evidence that I'd made a mistake, but that isn't meant as an excuse.*

Several people seemed to be worried by the amount of space needed by every process for the page tables for the tuple space. It's certainly a consideration, but the requirement is of the order of a few kilobytes, which is not enormous by modern standards. As it happens, giving the tuple space its own management system solves the problem anyway.

I accepted answers which concentrated on the need for exclusive access to tuples; this is one part of the general security problem, though perhaps it's close to protection.

*Some people thought that page table idea was good, but noticed that you'd need some sort of protection against unauthorised access. I don't think that anyone said how this could be done. Once you have an entry in the page table, that's just used by the memory access hardware whenever it's given an address to translate, which it does as quickly as possible; it isn't going to stop and check for special addresses. It **would** work in a fashion if you could somehow flag the page as protected in the page table, and the hardware was constructed to test the flag at every memory access and cause a supervisor call in case unauthorised access was attempted - but then you'd be no better off than with a supervisor call system anyway.*

*Not everyone has caught on to the idea that you can share memory between processes by putting the **same** real memory addresses in their page tables. Some objected to the proposed method on the grounds that every process's copy of the Linda memory would have to be changed with every Linda operation. I'm not inclined to accept that interpretation because an essential feature of Linda is to have a single shared memory (which is why I called it "the Linda shared memory"); ex hypothesi, you **can't** have a lot of copies of it.*

(d) If Linda is thought to be a good way to implement inter-process communication, then it can reasonably be expected that the operating system will support it, but on the grounds that the benefits of utility outweigh the costs of implementation. As well as that, the expense is limited, because the communication method isn't likely to be used all that often as compared with memory reference in general; it's more like an input-output operation than a simple reference to memory.

Not many people answered this part.

Some didn't seem to appreciate that it's quite hard work to operate such a memory. You need something like a segmented memory manager to allocate space for tuples, and some reasonably clever indexing scheme to find tuples of a given pattern. (You certainly don't want to have to search all the way through the tuple space every time someone makes a request.) Even an efficient implementation in a flat memory isn't going to run like lightning. The ideal implementation would be a large hardware associative memory, which I imagine would be extraordinarily expensive, so we have to simulate it in software. As usual, it costs a lot to implement a foreign memory model on unsuitable hardware.

Several people chose to answer a question something like "WHAT support should Linda expect from the system ?". As that wasn't the question I asked, any marks attached were accidents.

One good answer made the point that only the operating system was in a position to ensure the necessary level of security.

Alan Creak,
August, 1991.