UG Exam Performance Feedback
Second Year
2018/2019 Semester 1

COMP25111 Operating Systems

Jim Garside
John Latham

Comments Please see the attached report. Feedback from John Latham is awaited.

With some earlier marks as 'give-away' on coursework, the paper was intended to be a challenging test. The results statistics show that this was the case with marks spread across the range with something close to a 'normal' distribution: the overall average was slightly lower than might be desired. However, as the detailed comments show, there were a number of answers to questions which should have had easy parts (such as Q4 regarding Unix file permissions) which surprised the markers.

Questions were marked anonymously and (mostly) independently so there could be no influence of one good/poor answer from any candidate affecting judgement of other answers; the totals were only assembled when everything was complete.

- -

Q1: Explain the role of processor privilege mode(s) (as represented in the hardware) in running a secure operating system. [2 marks]

Largely a 'give-away' starter. There was a strongly held belief that there are exactly -two- privilege levels (satisfactory answer for this question) although there can be more (e.g. for hypervisor). There was some confusion with file access and general superuser privileges (managed in software). There is some idea that privilege levels protect different users from each other: this is not directly true - this is provided by the memory map not containing other users' addresses at a given time, not by controlling individual accesses to locations which are present.

There is a significant tendency to give highly non-specific answers, such as "prevents malicious accesses" without a hint as to what this implies or where the accesses are - or are not - made. Whilst this question has been marked generously it would be legitimate to require more specificity. For example, *processor* privilege prevents the completion of *instructions* which might violate *memory* locations (and some other operations, such as altering the interrupt enable status or mode changes); *file* access is not affected since the permissions will be checked in software.

Q2: Give an example of a part of a processor address space which should not be cacheable. Explain why this would be the case for your chosen answer. [2 marks]

Possibly a majority of candidates seemed to find an answer for this: as expected most chose (volatile) I/O space. An anomaly which occurred multiple times was "I/O space -for security-."; it's hard to see what people were thinking of here! There was a disconcerting number of irrelevant (and plain wrong) answers though. One recurring theme was

not caching the OS: note that the cache provides the data and any page protection operates independently. This was intended as a fairly easy question!

Q3: Given that a process can be in one of the three states [ready, running, blocked], which state transition cannot happen and why? [2 marks]

About half the class got the correct answer: "ready to blocked" cannot happen. A process cannot, by definition, become blocked unless it is already running. In remaining answers, the most popular impossible transition was "blocked to running", generally justified by saying that the transition would need to be "blocked to ready to running". This is not the case. A blocked process Q which has high priority can be unblocked by the scheduler and change state immediately from ready to running. In this case the process P that was running has been pre-empted by Q, and P's state will change from running to ready -- another transition that some people erroneously stated was not possible.

Q4: Files are a potential insecurity in a computer system; what measure(s) does a "traditional" Unix system use to control file access? Also, very briefly, outline a more flexible means of securing data in a filing system. [2 marks]

About half of the class were unable to explain the standard Unix rwx/ugo file permission model; this was what the question was asking for in the first part. This is worrying. For the second part, some students didn't answer it at all. Others described ACL or ACL-type systems where permissions are managed on a more fine-grained per-user basis, and got credit for this. Those who suggested setting passwords on files did not get credit.

Q5: What is meant by a "race condition"? Illustrate your answer with a short example. [2 marks]

This was generally well answered although some of the examples were not 'ideal' illustrations. It appears that the issue is generally understood.

Q6: Processes are isolated in their own contexts, but sometimes processes need to communicate with each other for various purposes. Choose four different ways processes might interact -- the more diverse the better -- name them and briefly describe what communications facility each might provide or when it might be appropriate to use it. [10 marks]

<No response supplied.>

Q7: Define what is meant by a computer "file system".

This was really intended as an easy introduction to a 'multi-part' question {Q7-9}; in retrospect, when broken up into Blackboard the question turned out to be somewhat ambiguous and answers varied according to candidates' interpretations. A liberal approach was therefore taken during marking and the average mark is consequently high.

Q8: Why might a computer support code to handle more than one file system implementation? [2 marks]

Really a follow-up to Q7 -- which it should put in better context -- and intended as a fairly easy thinking exercise: most students will have ported file devices from system to system for example. The average mark reflected this although there was some confusion between the file system architecture and the device organisation.

Q9: What are the advantages and disadvantages of running a file system's code in an unprivileged mode? In what type of operating system architecture is this done? [6 marks]

This question was reasonably well-answered, although a common misconception was that a filesystem running in an unprivileged mode would somehow lack the basic ugo/rwx kind of permissions necessary for the safe functioning of any filesystem. About 2/3 of the class realised that the question was referring to a "microkernel" architecture. The question was, of course, intended to merge the concepts of kernel architecture with the example of filing systems.

Q10: You need to run an image-processing application, which will take some time to complete, on a Linux desktop workstation. The application is capable of processing different sized images which are stored in disk files as pixel maps up to 10 MiB in size. You also have some other work to do on the computer whilst the image-processing utility is running. Outline the different stages the application is likely to go through from invocation to completion, highlighting instances of the different interactions between the application and the operating system and any background services the operating system is performing to support the application. [20 marks]

This question was intended to provide an opportunity for candidates to illustrate what they know about a range of operating system topics. There is a hint in "interactions" to think of the various system calls, interrupts etc. which might take place. A full answer should include operations on files, memory allocation, I/O - particularly DMA, process scheduling - with some thoughts of time-sharing and, particularly, priority of a background task, and some tidying up of resources and files after the event. There is scope to bring in other suggestions which could be credited where relevant: there are something like 30 marks worth of points which could be made (a max. of

20 would be awarded!) which means not every aspect needed to be recalled and described.

The first point to make - which was rather shocking - is that a 20 mark question - which ought to deserve ~30 mins. time in the exam - was frequently answered in a single, short paragraph of 3-6 lines; surely candidates have enough exam. experience to realise more than this would be required for a 'full' answer? Also, some candidates fixed on one or two aspects, such as scheduling, and laboured those rather than trying to encompass -many- aspects of an O.S. -- which is really the invitation in the question.

In technical details the best encompassed aspects were the DMA processes and, to an extend the scheduling, although prioritisation of processes was rarely raised and not always appropriate when it was. One might have expected more mentions of 'nice' or similar. File operations were mentioned although not usually described in much detail; attributes were occasionally checked for permission (okay) but not for file size, needed to allocate the space to DMA the data in to. Memory allocation tended to be confined to the start of the application, despite a strong(?) hint in the question that the data space would not be known.

All the expected aspects of the process were mentioned by one or more candidates, along with some (arguably) relevant issues which were not predicted; the latter were appropriately credited, of course. The disappointment stems from the small proportion of candidates who covered anything approaching the relevant breadth; too many confined themselves to a limited scope.

Many answers were highly generic, such as "Processes will be scheduled, perhaps by a 'round robin' algorithm." There is significant evidence of knowledge of the mechanisms in many answers but these are often just listed, rather than being applied to the question. E.g. "The scheduling algorithm could be A, B or C." rather than "The scheduling algorithm would be A because ...". To gain full marks the specifics of a question should be taken into account, not just a repetition of learned facts.

It is clear that many candidates have only a hazy idea of computer memory hierarchy - particularly the directly (processor) addressable space - mapped by an MMU - and the secondary storage for files et alia.

Q11: A multiprocessor computer system with a multiprocessing operating system is running several copies of an application which uses dynamic memory allocation. Any process may require a new page of physical memory, allocated from an adequately large, shared pool of free pages, at any time. What resource conflicts might occur and how might you design a page allocator to guarantee fault-free

operation? [4 marks]

Answers to this question were almost completely polarised. About half the class recognised that this question was about the conflicts of more than one processor trying to access the shared memory management data structures at any time. Thus some form of mutual exclusion mechanism needs to be provided to manage this. Those that recognised this generally explained it well and got full marks. The other half of the class did not address this conflict at all, and instead talked in general terms about memory allocation strategies. While in most cases this discussion was correct in itself, because it did not address the question, it scored no marks.

Q12: Why do operating systems typically include "device driver" abstractions rather than allowing direct access to peripheral devices? [2 marks]

Most students understood that the idea was to to make different hardware have the "same" interface to the OS, so the OS need not concern itself with the details of different varieties of hardware which provide a type of function, such as storage. A few students instead concentrated wholly on security, and that this was the sole reason for having device drivers.

Q13: Two Linux processes are using a block of shared memory. The programmer notices that the address of this memory appears at different addresses in the two processes, although the software works properly. How is this possible? [2 marks]

This was a straightforward question about virtual memory. Most students understood what was happening -- that each process has its own private virtual address space, and that memory mapping/address translation maps a process's virtual memory to real memory. In this case because the two processes have been written to specifically share a piece of memory (M), M exists in real memory, but it appears at different virtual addresses for each process. Some students seemed not to recognise what was happening, which is worrying.

Q14: In a high-performance computer with several layers of memory caching, the first level of the cache must be flushed on a process context switch. Why is this? [2 marks]

This question was generally very well-answered. Most people understood that the L1 cache contains process-specific data, so needs to be flushed before it is used for another process. Most people mentioned virtual addresses, but this was not necessary to get full marks, as long as the per-process idea was understood and explained.
