COMP25111

Two hours

Question ONE is COMPULSORY

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Operating Systems

Date: Tuesday 14th January 2014

Time: 14:00 - 16:00

Please answer Question ONE and any TWO other Questions from the other THREE Questions provided

---

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]

1. **Compulsory**

a) What is the key difference between a *system call* and a call to an ordinary method or function. Briefly explain why this difference is important. (2 marks)

b) In Linux, how does a shell implement a *pipe* between commands? (2 marks)

c) What does the term *starvation* mean in process scheduling? How may it arise? (2 marks)

d) Briefly explain the difference between a *process* and a *program*. What is the difference between a *process* and a *thread*? (2 marks)

e) What is the difference between a *monolithic* operating system and one constructed around a *microkernel*? (2 marks)

f) Direct memory access (DMA) is interrupt driven. Given that a processor writes to a disk, utilizing DMA, describe the four-step DMA process for writing data. (2 marks)

g) One method for implementing virtual memory is *paged virtual memory*. The procedure for this can be viewed as a sequence of steps. Explain the paged virtual memory procedure by outlining the sequence of steps for translating a virtual address to a physical address. (2 marks)

Question 1 continues on next page

Question 1 continued from previous page

h) Given that the purpose of a *page table* is to translate the page number (in the virtual memory) into a page frame (in the physical memory), and that the current partial view of the page table is:

[…]
[07, 00]
[06, xx]
[05, 03]
[04, 02]
[…]

where [X, Y] = [page number, page frame].

i) Calculate the page frames and page offsets given that the page numbers and page offsets to be sequentially translated are ‹07, 06› and ‹06, 01›, where ‹X,Y› = ‹page number, page offset›. (1 mark)

ii) Then state which page translation, ‹07, 06› or ‹06, 01›, causes a *page fault* (1 mark)

j) State how to avoid *external fragmentation* in memory. Your answer should include a brief description of the solution and a diagram supporting your description which explicitly covers the ‘before’ and ‘after’ scenarios. (2 marks)

k) What does the *dirty* bit in a page table entry indicate? State how it is utilised. (2 marks)

2. a) What is the difference between *preemptive* and *non-preemptive* scheduling? (2 marks)

b) Why is the size of the time slice in preemptive scheduling algorithms chosen to be significantly higher than the time taken for a context switch? (2 marks)

c) Explain briefly what a CPU burst is and what an I/O burst is. What is a CPU-bound process, and what is an I/O-bound process? Why is it a good strategy in process scheduling to give higher priority to I/O-bound processes? (4 marks)

d) Three processes A, B and C all alternate between fixed duration CPU bursts and I/O bursts, as follows. Process A has CPU bursts of 3 time units and I/O bursts of 4 units. Process B has CPU bursts and I/O bursts of 4 time units each. Process C has CPU bursts of 1 time unit and I/O bursts of 6 time units. Draw a diagram showing the states of these processes as they are run by a preemptive Round Robin scheduler for a total of 40 time-units, assuming that they all start ready at time-unit 0 and are queued in the order A (first), B, C (last), the time-slice adopted by the scheduler is 2 time-units, and the time for a context switch is negligible. For what fraction of the time is the CPU executing user processes? (5 marks)

e) A new scheduler is introduced using priority queues. There are two queues, Q1 (responsible for scheduling processes A and B) and Q2 (responsible for scheduling process C). Assuming they have ready processes, the two queues access the CPU alternately, as follows: Q1 gets 5 time-units, then Q2 gets 2 time-units, then Q1 gets 5 time-units, and so on. Processes in each queue are executed in Round Robin fashion with a time-slice of 2 time-units. If a queue runs out of ready processes before its allocated time-units have been used, it yields access to the CPU to the other queue. Apart from this, the situation is as described for part d) above. Draw a diagram showing the states of the three processes A, B and C as they are run by the scheduler for a total of 40 time-units. For what fraction of the time is the CPU executing user processes? (5 marks)

f) Briefly explain how *dynamic adjustment* of the priority of each process can be used to improve the behaviour of a scheduler. (2 marks)

3. a) To address the question: "given a single user program, where does it fit in memory?", state the steps an operating system must take in order to load a single user program into memory (this is termed *uniprogramming*). (5 marks)

b) In the context of converting an address generated by a program [a compiler] to the actual address, state:

i) The names of the two memory spaces involved; and (1 mark)
ii) The unit that performs [undertakes] this translation process. (1 mark)

c) With respect to *multiprogramming*, and the diagram in Figure 3.c, state the following:

i) How many programs can be loaded into the partitions in Figure 3.c? (1 mark)

ii) What operation is performed when all partitions are full and no more programs need to be loaded? (1 mark)

iii) What does the operating system normally do when a program performs an I/O operation? (1 mark)

iv) What happens when one of the programs finishes? (1 mark)

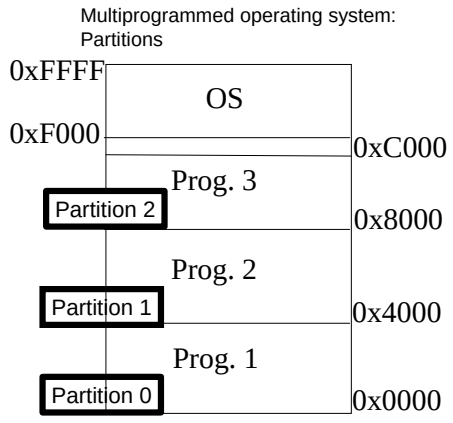

Figure 3.c. Typical diagram showing multiprogrammed operating system and associated partitions.

Question 3 continues on next page

Question 3 continued from previous page

d) Describe in detail how a *segment* is loaded; list the steps taken. (5 marks)

e) Given a physical address size of 2GB and associated 64KB block size, calculate the *number of page frames* in the physical address space. **NOTE**: To gain full marks you must show full working. (2 marks)

f) Given a 4GB address space and associated 16KB page size; calculate the *number of pages* that result in the virtual address space. **NOTE**: To gain full marks you must show full working. (2 marks)

4. a) Draw up a table that lists:

1. Page replacement policy name; and
2. Brief description of how the policy works.

In the table, describe three policies: First In First Out; Least Recently Used; and Not Recently Used. (5 marks)

b) Given that a keyboard sends characters into the computer system, name and describe the two registers normally used to undertake this process. (5 marks)

c) Explain what a *semaphore* is and describe the operations that can be performed on it. (3 marks)

d) In a certain system, the execution of three threads is synchronised using three semaphores, S1, S2 and S3, as shown below. Semaphores S1 and S2 are initialised to zero, while semaphore S3 is initialised to 1. All three semaphores are used only in the sections of code shown below.

Thread A	Thread B	Thread C
…	…	…
P(S1)	P(S2)	P(S3)
P(S2)	P(S2)	V(S2)
x=3*x	x=x+7	x=x-1
V(S3)	V(S1)	V(S2)
…	…	V(S2)
…	…	…

i) If the variable x is defined as an integer shared variable, initialised to 0, and is not assigned a value in any other sections of the code apart from those shown above, what will be its value when all three threads have finished executing? What will be the values of the three semaphores S1, S2 and S3? Justify your answers. (5 marks)

ii) Would the same final value for x be computed if the two P operations in Thread A were exchanged? Is there any other way in which the behaviour of the code might change? Justify your answers. (2 marks)

END OF EXAMINATION