

**SAMPLE NOT OFFICIAL  
EXAM**

Family Name:	
Given Names:	
Student Number:	
Regular Signature:	

**THE UNIVERSITY OF QUEENSLAND**

**School of Information Technology  
& Electrical Engineering**

First Semester Degree Examination **SAMPLE EXAM**, June 2004

**COMP3300/COMP7303**

**Operating Systems**

TIME: **TWO** hours for working

**TEN** minutes for perusal before examination begins

**ANSWER PART A ON SUPPLIED “True/False and Multiple Choice Answer Sheet” AND ALL OTHER QUESTIONS ON QUESTION PAPER**

**MAXIMUM MARKS = 50, 55 marks available**

**ANSWER PART A (30 MARKS) and B (5 MARKS) and ONE OF PART C OR PART D (each 20 MARKS) – THE COURSE COORDINATOR RESERVES THE RIGHT TO CHOOSE WHICH TO MARK IF BOTH SECTIONS C AND D ARE ANSWERED**

**OPEN BOOK:** written materials and a non-programmable calculator may be used but NO other computing or communication devices are allowed in the examination room.

Official Use Only									
Questions A1–A30 (computer-marked)	B	C1	C2	C or D	D1	D2	B–D TOTAL	marker	
/30	/5	/7	/13	D	/10	/10	/25		

**COMP3300/COMP7303 Operating Systems  
First Semester Sample Examination, June 2003**

**Part A – Everyone Must Answer: 30 marks – all questions 1 mark**

**multiple-choice: only 1 answer must be marked on the supplied “True/False and Multiple Choice Answer Sheet”; if more than one answer could apply, only the most accurate answer will be accepted**

**EXAMPLE**

- x. The answer to the fundamental question of life, the universe and everything is
- a the planet Earth
  - b 42
  - c unknowable
  - d it depends what the question is
  - e b and d

If you know your *Hitchiker’s Guide to the Galaxy*, **b** and **d** are correct, so **e** should be marked as your answer. Either **b** or **d** would be correct, but they are not individually the most accurate answer.

- A1. CPU speed for a given cost is improving by a factor of 2 every 18 months. DRAM cost is improving at a rate of a factor of 2 every 3 years. Which of the following statements is true, if it currently costs twice as much as you’d like to build a PDA to a given specification?
- a you must wait for CPUs to get cheaper
  - b you must wait for RAM to get cheaper
  - c you must wait for both but the CPU price will be the bigger holdup
  - d you must wait for both but the DRAM price will be the bigger holdup
  - e none of the above
- A2. You are designing a high-availability system. This means
- a you are trying to minimize hardware failures
  - b you are trying to minimize software failures
  - c you are trying to minimize downtime
  - d you are trying to maximize throughput
  - e both a and b but not c
- A3. In terms of requirements for systems programming
- a Java meets most of the requirements except low-level memory access
  - b Java’s automatic memory management is good for real-time programming
  - c Java’s support for primitive types like int would be useful for OS coding
  - d all of the above
  - e none of the above
- A4. When an interrupt occurs, the following has to be taken into account:
- a no registers can be used until state is saved by interrupt handling code
  - b some registers have to be saved by hardware before the interrupt handler can start
  - c all programs modify the program counter
  - d all of the above
  - e both b and c
- A5. A given interrupt handler should not be interrupted, and it is permissible that interrupts of this kind be missed. The following are options to avoid problems when this actually happens:
- a while state is being saved, interrupts are turned off
  - b an interrupt handler is reentrant (each instance uses different data memory)
  - c the device or program causing the interrupt retries if it’s not handled after a timeout
  - d b or c
  - e none of the above

**COMP3300/COMP7303 Operating Systems  
First Semester Sample Examination, June 2003**

- A6. Compared with a synchronous I/O request
- a an asynchronous request takes less time to process
  - b an asynchronous request takes longer to process
  - c the overall effect of an asynchronous effect is faster because I/O overlaps CPU usage
  - d both **b** and **c**
  - e none of the above
- A7. In general terms finding other work instead of waiting for I/O
- a makes the I/O operation faster
  - b makes the I/O operation slower
  - c is less an issue for I/O than keeping the CPU busy
  - d **a** and **c**
  - e none of the above
- A8. The Solaris approach to threads
- a emphasizes the ability of the operating system to control threads
  - b provides flexibility according to dynamically determined needs
  - c emphasizes the notion of threads in user space
  - d both **a** and **b**
  - e none of the above
- A9. The UNIX fork system call
- a can be efficient with copy on write
  - b is always an efficient way to implement parallel code
  - c supports the notion of threads in user space
  - d both **b** and **c**
  - e none of the above
- A10. For a round-robin scheduler
- a context-switching overheads favour short time quanta
  - b context-switching overheads favour long time quanta
  - c average wait time vs. preemptive SJF is usually longer
  - d both **b** and **c**
  - e none of the above
- A11. Initializing a semaphore  $S$  [algorithm p 201 Silberschatz *et al.* 2003; lecture week 5 slide 25] with a negative counter ( $N$ ) results in a situation where
- a a programmer made an error
  - b initially  $N$  resources are allocated, leaving none available
  - c  $N$  processes will be forced to wait before passing a *wait* call on  $S$
  - d if any process is forced to wait, any subsequent *signal* on  $S$  will restart it
  - e none of the above
- A12. Initializing a semaphore  $S$  [algorithm p 203 Silberschatz *et al.* 2003; lecture week 5 slide 28] with a negative counter ( $N$ ) results in a situation where
- a a programmer made an error
  - b initially  $N$  resources are allocated, leaving none available
  - c  $N$  processes will be forced to wait before passing a *wait* call on  $S$
  - d if any process is forced to wait, any subsequent *signal* on  $S$  will restart it
  - e both **b** and **c**
- A13. An atomic memory modification and test instruction
- a is essential for implementing synchronization
  - b is not needed for implementing synchronization
  - c makes implementing synchronization easier
  - d **a** and **b**
  - e none of the above

**COMP3300/COMP7303 Operating Systems  
First Semester Sample Examination, June 2003**

- A14. A simple spinlock (loop testing a variable, while attempting to set it atomically)
- a is generally efficient
  - b is always unsuitable for a single-processor system
  - c may be useful for a very short critical section with a low probability of contention
  - d none of the above
  - e both a and c
- A15. A deadlock will *never* occur if
- a no sequence of processes can be identified, each waiting for the next in sequence
  - b a process holding a resource has to release it before requesting another
  - c a process may not hold a resource unless it can be shared
  - d any of the above
  - e none of the above
- A16. Contiguous allocation in main memory is
- a always a good idea if you are running multiple processes
  - b always a good idea if you are running only one process at a time
  - c can lead to internal fragmentation
  - d can lead to external fragmentation
  - e none of the above
- A17. Segmentation in main memory
- a can lead to internal fragmentation
  - b can lead to external fragmentation
  - c is of no use for a single process
  - d b and c
  - e none of the above
- A18. In a situation where the physical address space is very small (compared to virtual)
- a an inverted page table will be efficient
  - b a 2-level page table will be efficient
  - c a simple forward page table will be efficient
  - d either a or b depending on memory usage
  - e b if each process uses most of its virtual address space
- A19. A multilevel page table
- a allows part of a page table to be paged out
  - b makes it less likely that the entire page table needs to exist
  - c in the worst case is slightly bigger than a simple forward page table
  - d all of the above
  - e none of the above
- A20. An optimal page replacement strategy is
- a inefficient at minimizing page faults but easy to implement
  - b efficient at minimizing page faults but almost impossible to implement
  - c efficient at minimizing page faults and easy to approximate
  - d similar to a clock algorithm
  - e none of the above
- A21. Storing swapped out virtual memory as normal files means
- a page organization on disk is more efficient
  - b memory-mapped files become easier to implement
  - c page tables need no special strategy to identify pages on disk
  - d b and c
  - e none of the above

**COMP3300/COMP7303 Operating Systems  
First Semester Sample Examination, June 2003**

- A22. Using a memory-mapped file differs from ordinary file I/O in that
- a** you use the same system calls in a different way
  - b** you need not read the file in explicitly
  - c** the programmer needs to explicitly write modifications back but not reads
  - d** **b** and **c**
  - e** none of the above
- A23. A tree-structured file system
- a** allows multiple items to have the same name
  - b** allows multiple names for the same item
  - c** prevents different users from using the same name
  - d** **a** and **b** but not **c**
  - e** none of the above
- A24. Aliases are a problem in file systems because
- a** keeping track of multiple items with the same name is complicated
  - b** dangling pointers can result from deletion
  - c** it is necessary to keep track of the number of names for each item
  - d** **a** and **b** but not **c**
  - e** **b** and **c** but not **a**
- A25. The SCAN disk scheduling algorithm
- a** attempts to minimize waiting time
  - b** can result in starvation
  - c** is organized to avoid looking at recently serviced parts of a disk
  - d** **a** and **c** but not **b**
  - e** none of the above
- A26. The C-LOOK disk scheduling algorithm:
- a** attempts to minimize waiting time
  - b** can result in starvation
  - c** is organized to avoid looking at recently serviced parts of a disk
  - d** **a** and **c** but not **b**
  - e** **b** and **c** but not **a**
- A27. A distributed operating system
- a** makes it explicit when you access a resource over the network
  - b** allows remote files to be used the same way as local files
  - c** is the same thing as a networked operating system
  - d** all of the above
  - e** none of the above
- A28. A malicious code fragment which can only run when attached to another code fragment is
- a** a worm
  - b** a virus
  - c** a Trojan horse
  - d** all of the above
  - e** none of the above
- A29. Language-based security (e.g., as in Java) makes sense if
- a** a language prevents insecure operations by design
  - b** most users can be trusted
  - c** the average user isn't a programmer
  - d** a language runs in a secure environment
  - e** both **a** and **d**

**COMP3300/COMP7303 Operating Systems  
First Semester Sample Examination, June 2003**

- A30. To implement secure code in C++
- a** is possible because of the design of the language
  - b** would require major limitations compared with standard C++
  - c** could be done by enforcing programming conventions
  - d** **a** and **c**
  - e** none of the above

sample

**COMP3300/COMP7303 Operating Systems  
First Semester Sample Examination, June 2003**

**B BONUS QUESTION FOR 5 MARKS (from various sections of the course)**  
i) Why are priorities useful in a scheduler? [3 marks].

ii) Explain why an inode-based multilevel pointer scheme does not require linear searches to find a given location in a file. [2 marks].

sample

**Part C – Answer *one of* PART C or PART D *not both* PARTS**  
**Design Issues: 20 MARKS TOTAL**

**Question C1 Threads and Processes–7 marks (from various sections of the course)**

- a. A proposed new operating system does not support threads kernel-level threads, but does have primitives to do the following:
- on an IO interrupt, restart a (possibly stalled) handler function in the same process as that which caused the interrupt; the API for such functions is:
    - // set up a handler routine to take control on an interrupt  
install\_handler (interrupt\_name, functionname);
    - // stall the handler, returning control to OS  
suspend ();
  - the handler routine replaces the OS's standard handling of the given interrupt once installed
  - such a handler, once installed, has its own stack and can access global variables; if it returns, the interrupt will be handled by the OS
  - generate an IO interrupt after a fixed time interval
  - allow a user-level program to define its own interrupt

Discuss how these features could be used to implement user-level threads. [4 marks]

SAMPLE

- b. Explain why threads are useful to obtain a responsive feel to a user interface. [3 marks]

**COMP3300/COMP7303 Operating Systems  
First Semester Sample Examination, June 2003**

**Question C2 General–13 marks (from various sections of the course)**

- a. The following is proposed as a solution to the bounded-buffer producer-consumer problem, with the intent that the whole buffer should be used (as opposed to the approach in the text book p 109; lecture week 3 slides 36–38) which leaves an entry unused:

```
initialized filled = 0, emptied = 0;

// if filled-emptied == size-1 all slots filled
// since arrays are indexed from zero
producer:
    while (true) {
        while (filled-emptied == size - 1) ; // spin
        buffer [filled % size] = produced;
        filled++;
    }
consumer:
    while (true) {
        while (filled-emptied == 0) ; // spin
        consumed = buffer [emptied % size];
        emptied ++;
    }
```

Comment on strengths and weaknesses of this solution. **[5 marks]**

S  
A  
M  
P  
L  
E

**COMP3300/COMP7303 Operating Systems  
First Semester Sample Examination, June 2003**

- b. For the FAT allocation scheme, under the following assumptions, calculate the size in bytes of a table needed to represent 64 files, each sized 1GB, on an 64GB drive:
- 1Kbyte block size
  - minimum bits to represent number of blocks in the table, plus 1 bit to represent end of file
- i State all assumptions and show detailed working [**5 marks**]

sample

- ii Discuss how your answer would differ with a UNIX-type inode scheme with direct, indirect double and triple-indirect pointers. [**3 marks**]

**Part D – Answer *one of PART C or PART D not both PARTS*  
Systems Programming: 20 MARKS TOTAL – based on assignments**

**Question D1 Multithreaded Code–10 marks**

In the following code fragment (part of a possible solution to Assignment 1):

```
// sort thread 2*i joins read thread i
void launch_threads (int N, char *fileNames []) {
    int i;
    FileInfo filesetup[N];
    ThreadInfo sortsetup[N];
    init_threads (2*N);
    sortsetup.N = N;
    for (i = 0; i < N; i++) {
        sortsetup[i].which_thread = 2*i;
        sortsetup[i].N = N;
        launch_thread (sort, (void*)&sortsetup[i], 2*i);
        filesetup[i].filemode = "r";
        filesetup[i].filename = fileNames[i+1];
        launch_thread (readFile, (void*)&filesetup[i], i);
    }
}
```

- a. Explain why there are timing-based bugs in this code. [4 marks]

- b. Rewrite the code to correct the problem(s). If you want to only show the parts which differ from the original, mark the places where changes are made in the listing at the top of the page, but write the changes below – *make it clear what you have done*. [6 marks]

**Question D2 Virtual Memory–10 marks**

Explain how the implementation of a FIFO policy differs from the implementation of a clock algorithm. Outline any differences in data structures and algorithms used.

sample

**COMP3300/COMP7303 Operating Systems  
First Semester Degree Examination, June 2003**

**Extra work space – make clear which question this relates to**

sample