THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester 2001

COMP2310
(Concurrent and Distributed Systems)

Writing Period: 3 hours duration

Study Period: 15 minutes duration

Permitted Materials: None

All your answers must be written in the boxes provided in this booklet. You will be provided with scrap paper for working, but only those answers written in this booklet will be marked. Do not remove this booklet from the examination room. There is additional space at the end of the booklet in case the boxes provided are insufficient. Label any answers you write at the end of the booklet with the number of the question they refer to.

Greater marks will be awarded for answers that are simple, short and concrete than for answers of a sketchy and rambling nature. Marks will be lost for giving information that is irrelevant to a question.

Name (family name first):

Student Number:

Official use only:

Q1 (25)	Q2 (25)	Q3 (25)	Q4 (25)	Q5 (25)	Q6 (25)	Total (150)

QUESTION 1 [25 marks]

(a) What does it mean for a real-time system to be *hard* or *soft*?

QUESTION 1(a) **[3 marks]**

(b) Explain the problem of *cache coherence* in a computer with multiple processors.

QUESTION 1(b) **[3 marks]**

(c) Some operating systems use device drivers with a lower half and an upper half. Which half is interrupt-driven and which is synchronous? Explain why. How do the two halves synchronize?

QUESTION 1(c) **[3 marks]**

(d) Describe what happens when the `fork()` system call is invoked.

QUESTION 1(d) **[4 marks]**

(e) Compare the operation of a token-ring network with that of a star network. Include discussion of transmission times and the causes and effects of failures.

QUESTION 1(e) **[4 marks]**

(f) Compare and contrast Unix processes with Java threads.

QUESTION 1(f) **[4 marks]**

(g) Define the *readers and writers* problem. Give an example of a typical operating system activity that is an instance of this problem.

QUESTION 1(g) **[4 marks]**

QUESTION 2 [25 marks]

(a) (i) One of the standard capabilities of a Unix shell is to run a program in the background. Usually the syntax involves placing an ampersand (`&`) at the end of the command line. Here's an example, with what I typed underlined:

```
richard@iwaki ls /usr/local/share/lib &  (then I pressed Enter)
[1] 19649
richard@iwaki CDE  Cshrc  Cshrc,v  xdm
      (at this point I pressed Enter again)
[1]    Done                          ls /usr/local/share/lib
richard@iwaki
```

Explain the meaning of each line of output. In particular, what is 19649?

QUESTION 2(a)(i) **[3 marks]**

(ii) Write two C fragments that you might expect to find if you search in the source code of the shell: one that shows in full how the shell runs a program in the background (i.e. what happened when I entered the command above), and one that shows what the shell does to detect the termination of background processes and clean up after them (i.e. what happened when I pressed Enter again). Your C code should be syntactically correct and it should use the various Unix system calls correctly.

QUESTION 2(a)(ii) **[12 marks]**

QUESTION 2(a)(ii) continued

(b) Some systems provide *general* semaphores; others provide only *binary* semaphores. The distinction between the two types is that the maximum value of the counter of a binary semaphore is 1: if you `signal` a binary semaphore when its counter is 1, it *stays* 1. Of course, you can also only initialize a binary semaphore to either 0 or 1. Suppose you have a system that only provides binary semaphores. Show how to implement general semaphores. That is, define a suitable abstract data type and provide full implementations of the general semaphore `init`, `wait`, and `signal` operations *in terms of* those provided for binary semaphores. (Do *not* attempt to implement general semaphores using lower-level operations.)

QUESTION 2(b) **[10 marks]**

QUESTION 3 [25 marks]

(a) Show how to use semaphores to achieve a rendezvous between two processes.

QUESTION 3(a) **[5 marks]**

(b) In any non-trivial operating system there are a number of places where clock interrupts (that prevent switching between processes) are temporarily disabled. Give an example where this is necessary. Is disabling such interrupts always *sufficient* for mutual exclusion? Explain.

QUESTION 3(b) **[5 marks]**

(c) Many spinlock implementations (e.g. ones that use test-and-set or compare-exchange operations) tend not to be *fair*. Explain what *fairness* means in this context. Explain *why* a spinlock implemented using test-and-set is not fair. (For this second part, I want a genuine explanation, not simply a restatement of what it means for the lock not to be fair.)

QUESTION 3(c) **[5 marks]**

(d) At most one process may be *actively* computing in any particular monitor. Explain what this means. Explain, using an example with condition variables, how several processes might have their program counters 'in' the monitor but why this does not count as *actively* computing.

QUESTION 3(d) **[10 marks]**

QUESTION 4 [25 marks]

(a) Compare and contrast the Ada rendezvous mechanism, occam channels, and the Linda tuple space. Your answer should indicate any restrictions on the communicating tasks, and whether each system is buffered or unbuffered, synchronous or asynchronous.

QUESTION 4(a) **[8 marks]**

(b) Define the term 'connection-oriented protocol'. Give an example of a standard network protocol which is connection-oriented.

QUESTION 4(b) **[3 marks]**

(c) Given a choice between using a message passing facility and an RPC facility, why might you choose to use the RPC facility?

QUESTION 4(c) **[3 marks]**

(d) List reasons why an RPC client might be unable to locate the server it wants to use. How might the RPC system deal with this possibility?

QUESTION 4(d) **[3 marks]**

(e) In lectures we examined four ways of dealing with RPC client crashes. List and explain each one briefly.

QUESTION 4(e) **[8 marks]**

1.

2.

3.

4.

QUESTION 5 [25 marks]

(a) Explain, with the aid of a diagram, what *nested transactions* are. Explain how each of the ACID properties applies to nested transactions. Pay particular attention to transaction committing and aborting.

QUESTION 5(a) **[8 marks]**

(b) (i) What is the fail-stop model of crashes?

QUESTION 5(b)(i) **[3 marks]**

(ii) What are idempotent and atomic operations?

QUESTION 5(b)(ii) **[4 marks]**

idempotent:

atomic:

(iii) What role do idempotent and atomic operations play in crash resilience?

QUESTION 5(b)(iii) **[4 marks]**

(c) (i) What are logging and shadowing?

QUESTION 5(c)(i) **[4 marks]**

logging:

shadowing:

(ii) How can logging and shadowing be used in an implementation of atomic operations?

QUESTION 5(c)(ii) **[2 marks]**

QUESTION 6 [25 marks]

(a) Describe, with the aid of an appropriate diagram, how MPI can be used to implement a worker/slave system that generates prime numbers. Your description should include the various MPI calls (including the appropriate parameters) and the tags you use. You do not have to write the whole program, but make sure that you do not omit any details related to the design.

QUESTION 6(a) **[10 marks]**

(b) Describe, with the aid of an appropriate diagram, how sockets in Java can be used to implement a connection-oriented protocol (such as in your `ceisius` system). Your description should include references to the various library calls involved. For each library call, which Unix system calls for sockets would you expect it to invoke?

QUESTION 6(b) **[10 marks]**

(c) Describe the wound-wait and wait-die approaches to distributed deadlock prevention. Which one can starve younger processes?

QUESTION 6(c) **[5 marks]**

Additional answers. Clearly indicate the corresponding question and part.

Additional answers. Clearly indicate the corresponding question and part.

Additional answers. Clearly indicate the corresponding question and part.