US202

Two Hours

UNIVERSITY OF MANCHESTER

INSTITUTE OF SCIENCE AND TECHNOLOGY

CT202 Concurrent Systems

For candidates taking:

BSc IN COMPUTING SCIENCE

SECOND YEAR SESSIONAL

MEng, BEng IN SOFTWARE ENGINEERING

SECOND YEAR SESSIONAL

MEng, BEng IN COMPUTER SYSTEMS ENGINEERING

SECOND YEAR SESSIONAL

MEng, BEng IN ELECTRONIC AND MICROELECTRONIC SYSTEMS ENG

SECOND YEAR SESSIONAL

MEng, BEng IN COMPUTING AND COMMUNICATION SYSTEMS ENG

SECOND YEAR SESSIONAL

Tuesday 5 June 2001 9.30-11.30

Answer **three** questions

The use of electronic calculators is NOT permitted

Note: Do not answer more than the required number of questions. Clearly cross out anything you do not wish to be marked.

PTO

1) Answer all parts

(a) Briefly explain the meaning of the terms mutual exclusion and critical section. By defining the wait and signal operations on semaphores and the use of an example, explain how semaphores can be used to provide mutual exclusion.

(6 marks)

(b) A small embedded system is responsible for gathering environmental data and transmitting it to a central computer system via one of *two* communications links. The software in the embedded system consists of four processes, each of which is responsible for gathering different types of data, temporarily storing it, and then transmitting it to the central computer system in blocks. In order to transmit data, a process must gain access to one of the communications links. Having gained access, it sends its block of data, and then releases the link. Each process may use either communications link, but once it has gained access to a link, no other process should be allowed to use that link until it is released. The bodies of the different tasks all share the following structure:

```
task body Tn is
begin
    loop
        while not finished gathering block
            get more data
        end while


        request access to a link
            use link
        release link
    end loop
end Tn
```

Using semaphores, devise appropriate protocols for requesting access to a link and releasing a link. *Your protocol should ensure that a process can only hold* ***one*** *link.* Briefly explain the purpose of any semaphores that you use, and define their initial values.

(8 marks)

(c) Consider a large relational database system that must be shared between many users concurrently. The operating system under which the database runs provides semaphores as the only synchronisation mechanism. When the database is to be updated, this must be done in mutual exclusion. Discuss the advantages and disadvantages of providing mutual exclusion:

- At the database level where a single semaphore protects the entire database
- At the table (or relation) level where each table is protected by a different semaphore
- At the level of individual records, where each record has its own semaphore

(6 marks)

2) Answer all parts

(a) The occam/CSP model of channel-based message-passing communication is said to be indirect, symmetric, and synchronous. Explain what this means. Briefly explain why a selective waiting/choice by channel construct is needed to support message passing communication.

(4 marks)

(b) Consider a sound system with two buttons (+ and -) for volume control. Button presses are converted into messages transmitted down two channels `louder` and `quieter` respectively. The channels are connected to a task that keeps track of the current volume (in an integer variable called `volume`). The task responds to messages on channel `louder` by incrementing volume, and to messages on channel `quieter` by decrementing volume. The new volume is then sent out on a channel called `to_amplifier`. Write the body of this task.

(5 marks)

(c) Briefly explain how condition synchronisation and mutual exclusion can be implemented in Java.

(5 marks)

(d) Develop a Java class that implements occam/CSP-style channels. Objects of this class should be able to transmit messages that are *single integer values*. The class that you develop should have methods `send` and `receive`, and may optionally have a constructor.

(6 marks)

3) Answer all parts

(a) What do places and transitions represent in a simple Petri Net model, and what is the marking of a Petri Net? Explain how a Petri Net model may be executed.

(8 marks)

(b) Discuss, in the context of Petri Nets, what is meant by the term reachability, and explain what a reachability tree is. What role does the reachability tree play in the analysis of a Petri Net?

(7 marks)

(c) With reference to the Petri Net shown below, develop the top 3 levels of the reachability tree (root and two further levels). What can you deduce about the properties of the Petri Net from your reachability tree?

(5 marks)

(Note: all arcs without an explicit weight have a weight of 1)

4) Answer all parts

(a) Part of the instruction set of typical processors is said to be *privileged*. Briefly discuss the role of the privileged instruction set in supporting the implementation of an operating system.

(3 marks)

(b) In the context of operating systems, what do you understand by the term *exception*. Briefly distinguish between exceptions and interrupts. How can the exception mechanism provided by most processors be used to provide entry to the operating system when application software makes a system call.

(5 marks)

(c) Consider three processes (p_1, p_2, p_3) executing concurrently on a machine that supports interrupt-driven input/output (I/O), and whose operating system utilises a priority-based pre-emptive scheduler. Each of the three processes begins by performing computations for a period ($t_{compute}$), and then performs I/O for a time t_{device}, after which each process terminates. All three processes use the same device for I/O. The values of priority, $t_{compute}$ and t_{device} for the three processes are:

- p_1: priority is highest, $t_{compute} = 20$, $t_{device} = 50$
- p_2: priority is medium, $t_{compute} = 30$, $t_{device} = 10$
- p_3: priority is lowest, $t_{compute} = 15$, $t_{device} = 35$

Draw a Gantt chart illustrating the behaviour of this system. Hence, or otherwise, determine the minimum time to execute this system. Explain your answer, stating any assumptions that you have made.

(8 marks)

(d) The operating system referred to part (c) is to be used in an embedded system. The application contains a process whose execution must be triggered by the occurrence of an interrupt from a device in the system's environment. Once the interrupt occurs, the process should be executed without delay. Explain how this could be achieved. State any assumptions that you make.

(4 marks)

5) Answer all parts

(a) Explain what is meant by the terms *contiguous* and *non-contiguous memory allocation*. Briefly discuss the operation of the first-fit, best-fit and worst-fit contiguous memory allocation schemes.

(5 marks)

(b) First-fit, best-fit and worst-fit schemes all require the memory manager to maintain a list of free-space (holes). This list could be sorted according to a variety of criteria. For these three schemes, identify the criterion (if any) that you would use to sort the free space list, and on the basis of the criterion, state how the list would be sorted.

(4 marks)

(c) At a particular instant of time, an operating system memory manager has the following list of holes (free space) in the following order: 250Kbytes, 120Kbytes, 470Kbytes, 390Kbytes. How would first-fit, best-fit and worst-fit memory allocation algorithms allocate memory to processes of the following sizes, arriving in the following order: 350Kbytes, 140Kbytes, 400Kbytes, 210Kbytes.

(3 marks)

(d) What are the disadvantages of contiguous memory allocation schemes, and how are they overcome by the use of paging?

(3 marks)

QUESTION 5) CONTINUED ON NEXT PAGE

QUESTION 5) CONTINUED

(e) Consider the following two C programs, which both simply set all the elements of the 64 by 64 array A to zero. Explain how the run-time pattern of memory accesses leads to page faults and determine:

- The number of pages needed to store the variables of program (i)
- The number of pages needed to store the variables of program (ii)
- The number of page faults that occur in the execution of program (i)
- The number that occur in the execution of program (ii)

Briefly explain your answers and state any assumption (besides those listed below) that you make.

Program (i)

```
main() {
    int A[64][64];
    int i, j;


    for  (i = 0; i < 64; i++) {
            for  (j = 0; j < 64; j++) {
                    A[i][j] = 0;
            }
    }
}
```

Program (ii)

```
main() {
    int A[64][64];
    int i, j;


    for  (i = 0; i < 64; i++) {
            for  (j = 0; j < 64; j++) {
                    A[j][i] = 0;
            }
    }
}
```

You should assume that:

- The page size supported by the operating system is 64 bytes.
- The program *code* occupies N pages exactly
- The operating system allocates N+1 pages to each program when it is first run.
- The array A is stored with adjacent elements within a row occupying adjacent memory locations, and the rows are stored one after another.
- An integer is stored in 4 bytes.

(5 marks)

END OF PAPER

----

p_1

p_2

p_3

p_4

t_2

t_1

t_3

t_6

t_5

t_4

2