UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1998

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
BEng Honours Degree in Information Systems Engineering Part II
MEng Honours Degree in Information Systems Engineering Part II
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER 2.5 / I2.5

OPERATING SYSTEMS II

Thursday, April 30th 1998, 2.30 - 4.00

Answer THREE questions

For admin. only: paper contains 4 questions

1 Consider a process-based operating system, using message passing as an inter-process communication primitive without buffering of messages. For this OS, processes are divided into two groups: *user processes* and *system tasks*, and the OS is equipped with a two-level scheduler. Each time a system task gets control of the CPU, it runs the assignment at hand to completion. This means that control may go to interrupt handlers, but the scheduler is not invoked when the process that had the CPU before the interrupt is a system task; instead, control goes as fast as possible back to the interrupted task.

a Is the code in the OS that does the actual scheduling part of a task or not? Explain your answer.

b Why could it be useful to introduce the distinction between user processes and system tasks (give at least three reasons).

c Explain how the fact that messages are not buffered can create problems for the clock and terminal interrupts.

d Describe the flow-of-control of a system call like `READ` that reads a block from disk. Make sure that you discuss the role of the interrupt handler, the system task, the user process and the scheduler.

The four parts carry, respectively, 15%, 15%, 30% and 40% of the mark.

2 a Discuss the differences between *synchronous* and *asynchronous* send, and their implications for the kernel.

b A car park (garage), with multiple gates that function both as entry and exit, has room for 100 cars and uses a computer to control the gates, using the followin processes:

- A `Gate` process for each gate that opens and closes the local gate to let a car in to or out of the garage.
- A `Detector` process (one for each gate) which sends a message to the relevant `Gate` process when a car arrives at the gate and when it has passed through the gate (so that the gate can be closed).
- A single `Controller` process which keeps count of the number of cars in the garage and decides whether to permit entry. The `Gate` processes communicate with the `Controller`.

If the garage is full, cars will queue at a gate waiting for another car to leave.

Using the following message passing primitives

`send (processname, message)` asynchronous send, sender is *not* blocked;

`receive (processname, message)` receiver blocks waiting for a message from process `processname`, if there isn't a message available already.

`processname = receiveany (message)` receiver blocks waiting for a message from *any* process, if there isn't a message available already.

give a pseudo-code outline implementation for the processes `Gate` and `Controller`.

The two parts carry, respectively, 20%, and 80% of the marks.

Turn over . . .

3 A kernel provides semaphore operations **down(s), up(s)** on a general semaphore s, and a call **install (inthandler, vector)** to allow a procedure to handle interrupts from a specified vector. A driver process for an analog-to-digital (A-to-D) converter device starts a conversion by setting a device-control register, when a request to read a value is received. When the device has completed the conversion it generates an interrupt. The driver reads the digital value from the device-data registers and writes it into an address specified in the request.

Assume a client requesting an A-to-D conversion places a request containing the following information on the driver's request queue:

clientsem – pointer to client's semaphore on which it waits for I/O completion to be signalled by the driver process,
response – address into which the disk driver must write the response (i.e. error or conversion value)

a Identify all semaphores needed for interaction between the driver and both client and interrupt handler. Give the initial values for the semaphores.

b Give an *outline* implementation of the client process showing how it interacts with the A-to-D driver request queue.

c Give an *outline* implementation of the A-to-D driver process and its interrupt handler. Assume the driver handles only one request at a time and does not perform retries for errors.

The three parts carry, respectively, 20%, 25% and 55% of the marks

4a A *tree structured directory* is typically implemented above a *flat* file system in many operating systems.

i) Explain what is meant by a *tree-structured directory* and what is meant by a *file path name*.

ii) Describe the information stored in both the tree-structured directory and the flat file system.

iii) Describe how a path name is translated into a physical disk address.

b A real-time system provides an interrupt handler process **I** which must deal with an interrupt inter-arrival time of 25 ms and requires a maximum service time of 5 ms. There are four additional processes:

Process	Period (ms)	Maximum Execution Time (ms)	Required Priority
A	100	25	4
B	75	15	3
C	100	20	5
D	150	30	2

Note: lower numbers indicate higher priority.

i) Briefly explain the *rate monotonic scheduling* strategy

ii) Show that not all the above processes can be members of the *set of schedulable processes*. With the aid of a diagram, show which processes are members and show that these processes meet their deadlines.

The five parts carry, respectively, 15%, 30%, 15%, 10% and 30% of the marks.

End of paper