UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1996

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

Friday, May 3rd 1996, 2.00 - 3.30

Answer THREE questions

For admin. only: paper contains
4 questions
4 pages (excluding cover page)

1 A mailbox is a data structure which is used by processes for exchanging messages. The mailbox is implemented within the kernel so is independent of both senders and receivers. Assume each mailbox can buffer up to 10 messages where a message has a fixed length of 100 bytes. The following data structure is be used, within the kernel, to represent a mailbox:

```
TYPE message = RECORD
        next : pointer to next message in queue i.e. array index number
        free: boolean
        msg: packed array  [1..100] of byte
END

Mailbox = RECORD
        msgq: queue          -- queue of sent messages
        receiverq :queue     -- queue of  blocked receivers
        count : cardinal     -- of free messages
        buffers: ARRAY [1.. 10] OF message
END
```

The following mailbox operations are implemented within ***the kernel:***.

SEND (VAR mb:mailbox; VAR smsg:message ; VAR result:resultcode)
- an asynchronous send so the messages must be buffered by kernel.
- resultcode = OK or no_buffers in mailbox

RECEIVE (VAR mb:mailbox; VAR rmsg:message)
- the receiver is blocked until a message is available.

Give *pseudocode* implementations for the SEND and RECEIVE kernel procedures specified above.

Assume the kernel executes with interrupts disabled. There is no need for detailed queue manipulation code e.g. pseudocode of the form:

```
insert at end of queue X
remove from front of queue Y
```

is acceptable. Your pseudocode can ignore type checking.

2 A computer system controls a number of printers, but does not implement any spooling. When a client process requires a printer, it must queue a *request* record on the printerQ, and then block waiting for a printer to become available. After printing a file, the client process must indicate the printer it had been allocated is now free, by queuing a *release* record on the printerQ, and can continue processing. Assume the computer system provides a call **Print (printerid, filename)** to print a file on a specific printer

The computer system supports the semaphore operations **P (s)** and **V (s)** where **s** is a pointer to a general semaphore.

a Identify all the semaphores that are needed to control the interaction between the multiple client processes and the printer manager process, as well as their initial values. Describe the parameters that must be passed in the *request* and *release* records.

b Give *outline pseudocode* for a client process and for the printer manager process. Indicate how the printer manager holds information about free printers and clients waiting for a printer. The pseudocode for queue manipulation can be of the form indicated in question (1), above.

The two parts carry, respectively, 35%,and 65% of the marks

Turn over

3a Outline a strategy which would permit users to share open files, describing:

i) The information held in memory for each open file.
ii) The constraints which must be satisfied on opening a file for reading or writing and information which must be maintained to prevent interference between readers and writers.

b A real-time system provides an interrupt handler process **I** which must deal with an interrupt inter-arrival time of 50 ms and requires a maximum time of 10 ms to service the interrupt. There are four additional processes:

Process	Period (ms)	Maximum Execution Time (ms)
A	200	40
B	150	40
C	200	50
D	300	70

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.*

iii) Assume that both A and D are high priority processes which must be kept in the set of schedulable processes. Explain how this can be achieved within the rate monotonic strategy. With the aid of a diagram, show that the members of the set meet their deadlines.

The five parts carry, respectively, 25%, 25%, 10%, 10% and 30% of the marks

Turn over

4a Assume an operating system supports the following message passing primitives:

> **send** (destination: procid, VAR msg: contents) - an asynchronous send, where the sender continues after sending message msg to destination process
>
> **receive**(source: procid, VAR msg: contents) - the receiver is blocked waiting for a message from the source, if the message is not available
>
> source := **receiveany** (VAR msg: contents) - the receiver is blocked waiting for a message, from any process, if none is available. Source indicates the sender of the message.

The system also provides a call install (inthandler, vector) to allow a procedure to handle interrupts from a specified vector.

A serial printer which generates interrupts for each character printed is controlled by a printer-driver process. This process receives and prints event messages consisting of a string of up to 80 characters from client processes in the system. Give a *pseudocode* outline of the printer-driver and its procedure to handle interrupts.

b The computer is connected to a network and a user process can send or receive messages to processes in remote computers over the network, using the primitives described in (a) above. Briefly explain the operating system functions, with respect to process addressing, what buffers are required and when user processes are blocked in order to implement the message primitives over the network. Discuss whether it would be feasible to adapt a disc-driver process as a network-driver process.

End of paper