

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE
UNIVERSITY OF LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2000

ISE PART I: M.Eng. and B.Eng. and ACGI

OPERATING SYSTEMS 2

Thursday, May 11 2000, 2:00 pm

There are FOUR questions on this paper.

Answer THREE questions.

All questions carry equal marks.

Corrected Copy



Time allowed: 1:30 hours

Examiners: Dr S.V. Van Bakel

- 1
 - a Is the code in the OS that does the actual scheduling part of a task or not? Explain your answer.
 - b Explain how the scheduler works, how its gets activated, and what activates it. Give a diagram that describes the Round Robin Scheduling Algorithm.
 - c Give the *process state diagram*. Give an extension that deals with the notion of a process being *swapped out*, i.e., the process descriptor exists, but that no longer has its code in memory.
 - d Describe in a few lines the effect of the Unix system calls `fork` and `exec` and how they interact.

Try to be to the point for each case. 100 words per part should be the maximum.

The four parts carry, respectively, 15%, 30%, 30%, and 25% of the marks.

- 2 a Describe, using a diagram, in which way the blocks used for a file are administered within a MINIX inode. Give at least three other data contained in an i-node located on disk. Give also the data that are kept extra in i-nodes for open files.
- b Give the names of at least four disk scheduling algorithms and describe them in a few words.
- c Describe via a picture the typical file sharing model. Give a short description of what is kept for each entry in the tables involved.
- d Discuss the UNIX/MINIX file sharing model. In your answer, describe the data management by the OS when a file is opened; make sure to discuss the various file tables involved (you can ignore the i-node handling).

The four parts carry, respectively, 25%, 20%, 30%, and 25% of the marks.

Turn over ...

- 3 a Describe each of the following access control methods, and explain how they are used (for each use no more than 30 words)
- Protection Domain
 - Access Matrix
 - Access Control List
 - Global Table
 - Capability
- b Discuss how capabilities can be revoked in a distributed system.
- c The following access matrix is used to control access to resources in a system

	File1	File2	File3	Printer	D1	D2	D3	D4
D1	<i>read, write+</i>					<i>enter</i>		
D2	<i>read</i>	<i>write</i>	<i>owner</i>					
D3	<i>read</i>	<i>read</i>	<i>exec</i>	<i>owner, print</i>				<i>enter</i>
D4		<i>exec</i>	<i>read</i>	<i>print*</i>		<i>control</i>		

The following are members of the domains:

- D1 - Charles
 D2 - Mary, Fred
 D3 - Fred, James, Joan
 D4 - Pat

Explain how you are able to derive the answers to the following questions:

- i) Who can write to File1?
- ii) How can Charles print to the Printer?
- iii) Identify all resources and operations accessible to Fred.
- iv) Can Mary lose the right to write to File2?

The three parts carry, respectively, 30%, 40%, and 30% of the marks.

- 4 a Give, in a picture, the structure of process queues in MINIX, and discuss how the scheduler deals with them.
- b In MINIX, both asynchronous send and synchronous send are used for message passing. The message is sent asynchronously if the receiver is not waiting for a message, but has no message pending; one message can be stored for each process. Otherwise, if it wants to send to a process that has already a message pending, the sending process blocks. Explain how messages still can get lost.
- c In the process administration of the Unix-like operating system MINIX, process 0 (zero) corresponds to the pseudo-process `HARDWARE`, that serves as the source of the messages which are sent as the direct result of an interrupt. Although this pseudo-process has a number, it does not have a slot in the Process Table (hence the predicate 'pseudo'). Explain.
- d In MINIX, system calls are implemented using `SendRec`, forcing the user process to explicitly wait for the result from the requested action to arrive. Kernel processes, however, can send messages without having to wait explicitly for an answer. Although this sounds reasonable in the communication between user processes and the kernel, it is as such not a safe protocol, since deadlocks can occur. Explain why this creates no problem in the communication between user processes and the kernel, and how the organisation of the message passing between tasks in MINIX avoids deadlocks.

The four parts carry, respectively, 20%, 20%, 20% and 40% of the marks

End of Paper