Computer Science 34?

(for 415.340 and 415.341)

Operating Systems

TEST, 1997

READ THIS FIRST !!!!!!!!!!!!!

Answer all questions -
- BRIEFLY. I don't think that you should need more than one side of paper for any of the questions; adjust your answer to fit.

Make sure your name is on every piece of paper which you hand in.
The same number of marks is allocated for each question.
The marks for each question are equally divided between the parts of the question.
There are FOUR questions; the test lasts for 90 minutes. As I imagine you can work out for yourself, that can reasonably be interpreted as about $22^1/_2$ minutes for each question.
I expect answers in terms of the material covered in the first half-semester of the 34? course, augmented by such general computing knowledge as can reasonably be expected of a stage 3 Computer Science student; dissertations on topics not treated in that part of the course are not required and will not receive any marks.
I try to give all the information you need to answer each question, but don't always succeed. If you consider that you have insufficient information to answer a question, don't ask a question in the test : explain your difficulty in your written answer, say what further information you would need to resolve it, and make clear any assumptions that you have made.

READ THIS NEXT :

All the questions in this test are about a simple sequence of operations. This is the sequence :

Step 1 : An appropriate editing programme called P is used to display a file X.
Step 2 : The programme is stopped.
Step 3 : The file X is deleted.

You can assume throughout that the file system is Unix-like.

The symbol <ret> denotes the "return" or "enter" key.

QUESTION 1.

What happens at the user interface when Step 3 is performed using

(a) a Macintosh-like interface, in which deletion is accomplished by using a mouse to drag an icon to another which means "delete";

(b) a Unix-like interface, in which deleting a file called X is accomplished by typing "delete X".

In each case :

(i) state the actions taken by the user;
(ii) describe the signals which reach the system;
(iii) explain what the system does with the signals;
(iv) explain how the system identifies the action to be taken (that is, "delete") and the name of the file.

Do NOT describe how the instruction is executed, nor how the file is located.

NOW TURN OVER THE PAGE !!!

QUESTION 2.

Explain how the files containing the programme P and data X required for Step 1 are identified in the two cases :

(a) with the Macintosh-like interface, when the X icon is double-clicked;

(b) with the Unix-like interface, when the instruction "P X<ret>" is typed.

Do not describe how the double-click is identified, nor how the characters of the instruction are read; start from the point at which (a) it is known that a double-click has been detected at a point on the screen with known coordinates, or (b) with the system ready to receive input characters, and an empty system buffer in which they can be stored. Ignore all matters connected with protection and security.

Identify the information which the system must have to complete the task successfully, but ignore fine detail, such as character-by-character comparisons, arithmetic carried out on screen coordinates, etc..

You may assume that the operation is eventually completed successfully.

QUESTION 3.

Assuming a Unix-like protection-code system (ignoring the group level), with read and execute privileges identified for owners and non-owners of files, answer these questions for Step 1 :

(a) When are the protection checks performed ?

(b) What information does the system need to perform the checks, and where does it comes from ?

(c) What are the messages (if any) which you would expect the system to display in all possible combinations of forbidden or allowed access to both of the files in both of the systems ?

QUESTION 4.

Describe how memory is allocated during Step 1, and released during step 2. Consider two sorts of memory management system :

(a) A primitive system designed to run just one programme, but with upper limit and lower limit addresses provided;

(b) A segmented memory management system, with an API function which allocates memory blocks of sizes requested, returning the first available address of the block, or zero if the request is refused, and another function which releases a block given its initial address.

Ignore details of displaying the file, reading the file, and things that P has to do to address the material in memory.

Alan Creak,
September, 1997.