Computer Science 340

Operating Systems

FIRST TEST, 1993

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 THREE questions; the test lasts for 45 minutes. As I imagine you can work out for yourself, that can reasonably be interpreted as about 15 minutes for each question.

I expect answers in terms of the material already covered in the 340 course, augmented by such general computing knowledge as can reasonably be expected of a stage 3 Computer Science student; dissertations on topics not yet treated in 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.

QUESTION 1.

In order to edit a file called QUESTION1 -

- on a system running a Unix-like shell, you might enter the text

```
vi QUESTION1
```

at a keyboard, and press the RETURN key;

- on a Macintosh system, you might move the screen pointer to lie on the icon for the file QUESTION1, and double-click.

Assuming (realistically) that the underlying operating systems are broadly similar, both these sequences of instructions must reduce to something like this basic sequence :

```
locate the editor;
locate the file to be edited;
execute the editor, giving it the file to be edited as a parameter.
```

For each system, identify the operations which MUST be executed to transform the computer input into a reasonable equivalent of the required sequence.

I do not want a full description of ***how*** *all the operations are implemented; I want two reasonable sequences of events which will transform the input received by the computer into the required sequence. They could perhaps be presented in terms of a sort of pseudocode.*

I suggest that you try designing each procedure from the top down, and stop when you get to anything specific to a particular system.

(CONTINUED ON THE NEXT PAGE.)

QUESTION 2.

This question is mainly about *short cuts*. Here are two examples to begin with.

Unix example.

In a Unix system with a conventional shell, one sometimes enters rather long instructions. For example, to delete a file addressed as A/B/C/D/E from your working directory, you enter

```
rm A/B/C/D/E
```

If you make a mistake, you have to reenter the whole instruction - and, in doing so, you may well make another mistake. The shell therefore provides an easier way : you can edit the line you just entered and submit it again. Here's an example, using a rather short line for simplicity's sake :

```
% ls
deadone    deadthree   deadtwo    liveone    livethree   livetwo
% rm deaf*
No match.
% ^f^d
rm dead*
% ls
liveone    livethree   livetwo
```

Macintosh example.

In a Macintosh system, to delete document E in folder D in folder C in folder B in folder A which you can see open on the desktop, you open B by double-clicking its representation in A, then open C ...... then delete E by dragging its representation in D to the "Trash" icon. You are left with folders B, C, and D open on your desktop, which is a nuisance if you don't want them any more.

The Macintosh system therefore provides a way to avoid the clutter : if you hold down the "option" key when you open an object, the parent folder is closed as the object is opened.

Using these examples as illustrations, answer these questions :

(a) How do these short cuts contribute to the "system genies" of their respective systems ? *(You may like to show how they help to produce a "simple, small, and powerful" system genie.)*

(b) Are the short cuts "intuitively obvious" ? Should they be "intuitively obvious" ? How should people be informed of them ?

(c) Here's a little more of the Unix example :

```
% ls
deadone    deadthree   deadtwo    liveone    livethree   livetwo
% rm dead(
Too many ('s.
% ^(^*
rm dead *
rm: dead nonexistent
% ls
%
```

(NOTE that the shell instruction "`rm A B C ...` " deletes all the files in the list following `rm`.)

Comment on this as an example of interface design and implementation.

(CONTINUED ON THE NEXT PAGE.)

QUESTION 3.

NOTE : In this question I ask you for a large number of fairly specific responses. Do not write essays. If each part of your answer has a fairly sensible justification, which can usually be given in a few words, I'll be fairly happy.

A shared computer system must maintain information about all those who use it. The collection of such information is sometimes called the *user database*. Various protection and security measures are necessary for the reliable operation of the user database, and these may be quite different for different items of information recorded.

This question is about the protection and security requirements for three sorts of item : the login password, the search path, and the accounting information.

For each of the three items, answer the two questions below, *giving a brief justification of your answer in each case*. (The justification should explain both why certain access modes are not permitted and why other modes are.) Any answers or parts thereof which apply to more than one of the items should be clearly identified as such.

(a) State who or what should have access to the item, and what sort of access should be permitted.

(b) Suggest how your recommendations should be implemented. Comment on whether the item should be stored separately from other items (for example, in a special password file); whether the protection mechanism should be subject-based or object-based; and what mechanism you would recommend (for example, file protection codes, passwords, supervisor calls).

Alan Creak,
April, 1993.

(CONTINUED ON THE NEXT PAGE.)