Computer Science 340

Operating Systems

FIRST TEST, 1992

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

Answer all questions.
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 number of asterisks (*) shown against each part of a question is proportional to the number of marks allocated to that part within 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.

> Questions 1 and 2 are about different topics; one of them is about graphical and textual interfaces, while the other is about different ways of presenting instructions to the operating system. Do not confuse them !

** (a) It is claimed that :

(i) graphical interfaces for operating systems are more self-explanatory than textual interfaces;
(ii) it is easier to select an action using a graphical interface than with a textual interface.

Comment on both these assertions, using the Macintosh and Unix systems as examples if you need them.

* (b) Graphical User Interfaces are all very well, but they are less effective if you can't see them. Some recent work has been directed towards implementing "Soundstation", a two-dimensional auditory equivalent of a desktop in which objects are identified by sounds rather than by sight. The system is intended for use by people with seriously impaired vision (A. Karshmer, R. Hartley, K. Paap : "Soundstation II : using sound and sound spaces to provide high bandwidth computer interfaces for the visually handicapped", *Sigcaph Newsletter* **#44**, 1 (January 1992)). Instead of the graphical interface's icons, it is proposed that there should be "earcons", characteristic sounds associated with different positions in the space, representing things in the system and subject to manipulation in the same ways as icons. Input is to be through conventional mouse and keyboard. When the "pointer" is on an "earcon", an appropriate tonal cue is generated; the "earcons" otherwise remain silent. Assume that the pointer position is shown at regular intervals by a short beep. (Do not worry about the technical problems; an appropriate arrangement of four loudspeakers can be operated to provide effective two-dimensional stereophony.)

Consider your answers to (i) and (ii) above, and in each case compare the properties of the "Soundstation" interface.

QUESTION 2.

> Questions 1 and 2 are about different topics; one of them is about graphical and textual interfaces, while the other is about different ways of presenting instructions to the operating system. Do not confuse them !

* (a) Explain what is meant by "metaphor", as it is used in the terms "desktop metaphor" and "tools metaphor". Why are such "metaphors" (alternatively called "illusions") now thought to be important after many years during which operating systems were designed without worrying about them ?

** (b) Those who favour the desktop metaphor suggest that it is "more natural" to describe a task by identifying the object - usually a file - which is to be worked on during the task than by specifying the tool - usually a programme - which is to be used. Here are two (for political reasons) descriptions of jobs written by people who were, presumably, not worrying too much about system metaphors. Discuss the descriptions *briefly*. Do the authors use the tools metaphor, the desktop metaphor, or some other ? (Or, perhaps, a combination of several metaphors ?) Comment on any implications of your conclusions for operating systems design.

(i) *Edmonds cookery book* (T.J. Edmonds Ltd., De Luxe edition, 16th printing, 1978), page 80.

1 Beard and drain the oysters;
2 dip in flour and a little pepper, mixed.
3 Beat an egg;
4 dip in oysters;
5 coat with breadcrumbs.
6 Put dripping in frying pan and
7 when smoking hot
8 put oysters in and
9 fry a golden brown.

(ii) *Practical home woodworking illustrated* (Odhams Press Ltd., 5th printing, 1955), page 296.

1 In gluing up the carcase,
2 glue in the rails first, and
3 leave the cramps in place
4 until the glue has set.
5 Temporarily put the top and bottom in place, and
6 test across the front from corner to corner for squareness.
7 When gluing in the top and bottom,
8 glue only the front 4 in.,
9 remembering that the wood will probably shrink in the course of time,
10 while the width of the framed ends will not.

(CONTINUED ON THE NEXT PAGE.)

(NOTES :

I have added the line numbers for easy reference, and the indenting.

in (i) :

in line 1, "Beard" means "Remove the oysters' beards". It doesn't matter what the beards are.
line 4 means "dip the oysters in the egg", not the other way round.
in line 6, "dripping" means "a sort of fat for cooking"; it does not describe the condition of the oysters.

in (ii) :

in line 6, "squareness" means "perpendicularity".
in line 8, "in." is short for "inch", a mediæval unit of length approximately equal to 2.54 cm., and now used only in backward areas such as the U.S.A.
"Carcase", "rails", "top", "bottom", and "ends" are parts of the object under construction. (A filing cabinet.)
"Cramps" are tools used to hold parts together during construction.

)

QUESTION 3.

NOTE : In both parts of this question, assume :
- that all programmes run in a single computer which has a single processor;
- that you have full access to any system manuals which you require; and
- that security is identified with the inability of any non-system programme to gain any sort of uncontrolled access to the memory owned by another programme present in the system.

** (a) It is claimed that hardware assistance is needed properly to implement a protection system in any computer shared by two or more people. Two hardware features are commonly provided for this purpose :

- memory address checking, typically provided by base and limit registers or some equivalent, and

- a processor supervisor mode in which certain privileged machine instructions are available, usually entered by a supervisor call operator which effects both a branch to a standard operating system procedure outside the programme's address range and a switch from normal to supervisor mode.

Consider the security of systems with only one of these features. In both cases, state whether security is possible, either by demonstrating how it can be provided or by explaining how to subvert security. (Bear in mind that the system must continue to be sharable; schemes in which one programme monopolises the computer for ever are not acceptable.)

FOR BONUS MARKS EQUAL TO FULL MARKS FOR THE TEST : Explain a foolproof method which is guaranteed always to subvert security if both features are present. Your method must operate when encoded in a programme which is executed as a normal programme in normal mode; it must not depend on any physical interference with the system, or on the actions of any person operating the system. NOTE : If I can find a flaw in your method, you will - of course - not receive the bonus marks, and I shall impose a penalty equal to full marks for the test.

* (b) With the early operating systems which implemented partitioned memory, several programmes could occupy memory simultaneously, each in a different area known as its partition. Neither of the features mentioned in part (a) was present; security depended entirely on the discipline observed by the programmes.

You have access to a system of this sort only through a trusted* compiler which is known to produce perfect code including enough software checks to guarantee that when the code is executed no memory reference is ever made to an address outside the specified partition. Show how you can gain access to any part of the computer's memory.

Alan Creak,
April, 1992.

* - I originally mistyped this word as "rusted". It put quite a different aspect on the question.