Computer Science 340

Operating Systems

TEST, 1998

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 340 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.

QUESTION 1.

(a) When a process sets up and uses a stream for communication with other entities (devices, processes, etc.), it must send information to, or receive information from, (or both) other parts of the system. Describe the sorts of information which must be sent or received if the communication is to be correctly managed.

(b) A process opening a data stream issues a request equivalent to

open(disc, "filex")

in which "disc" identifies a device and "filex" is the name of a file present on the device. Describe the actions taken by the operating system to open the stream, and the structures set up during this operation.

(c) In many operating systems, a new file is not entered into the disc file table until, and unless, the process which created it executes some form of **close** instruction. Explain how it is possible to use a file which does not appear in a directory.

QUESTION 2.

It is claimed that hardware assistance is needed to implement an effective 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 both executes a branch to a fixed machine address and switches the processor from normal to supervisor mode.

Assume in your answer that the processor in the system you are discussing has no other unusual characteristics.

(a) To what extent are these features necessary in protecting a process's primary memory from interference by another process ? (Bear in mind that the system must continue to be sharable; schemes in which one process monopolises the computer for ever are not acceptable. Also observe the "necessary" isn't the same as "desirable".)

(b) What additional considerations must be taken into account when considering protection of files ? To what extent can the hardware features mentioned above be used in protecting disc files against access by processes not belonging to their owners ?

(c) In justifying the use of encryption rather than hardware as a basis for protection systems for material held in disc files, it has been argued that it doesn't matter whether or not an intruder can read other people's material provided that the material is impossible to decode. Is that true ? (– and give reasons !)

QUESTION 3.

If you want to edit a file called QUESTION3 –

– on a system running a Unix-like shell, you might

(i) enter the text "vi QUESTION3" at a keyboard, and
(ii) press the RETURN key.

– on a Macintosh system, you might

(i) cause the screen pointer to lie on the icon for the file QUESTION3, and
(ii) double-click the mouse button.

Apart from the user interface, the underlying operating systems are broadly similar, so the same operations must be executed by the operating system in both cases to prepare the file for editing.

(a) Leaving out tasks of interpreting the input and preparing the screen display, what basic operations must the system perform to carry out the instructions given ?

(b) State the information which the system must deduce from the input it receives in order to carry out the operations you identified in (a).

(c) For the Unix system, describe the signals which reach the system when you use the interface as described above.

(d) For the Macintosh system, describe the signals which reach the system when you use the interface as described above.

(e) For the Unix system, explain how the information can be derived from the signals received.

(f) For the Macintosh system, explain how the information can be derived from the signals received.

In (e) and (f) 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 information. They could perhaps be presented in terms of a sort of pseudocode.

Do not give full details of actions taken by system components not closely connected with the interface management software. (For example, assume that the file system can locate a file given its name.)

You might find it helps if you try to be precise in your answers – for example, you do not move the pointer and click on an icon; you move the mouse and click the mouse button.

(NOTE : You are not expected to know the detailed mechanisms of the Unix and Macintosh systems.)

QUESTION 4.

A simple memory manager allocates segments of memory on request. Each segment in memory has a header which includes (among other things not relevant to this question) the length of the segment and a flag showing whether or not the segment is in use. Two operations are provided :

get : size → pointer	*either* a segment of the specified size is allocated using a cyclic first fit algorithm, and a pointer to the segment is returned to the calling process, *or* the operation fails;
release : pointer →	the flag of segment identified by the pointer is set to "not in use"

(NOTE : the notation A : B → C means that function A given a parameter B returns a result C to the process in which it was executed.)

There is no system memory map, and no provision for moving segments about in memory. The memory manager's knowledge about the current state of storage allocation is a pointer to the address following the end of the segment most recently allocated.

First explain how the memory manager can implement the **get** and **release** functions –

then illustrate your answer by describing how a system beginning with an empty memory of size 10 responds to this sequence of operations :

S1 = get(4);
S2 = get(4);
release(S1);
S3 = get(3);
S4 = get(2);
release(S2);
S5 = get(3).

Alan Creak,
September, 1998.