Computer Science 34?

Operating Systems

TEST, 1997 : Answers

These are not the only possible answers, nor necessarily the best ones. They are not always the shortest ones; I have sometimes preferred to present rather fuller explanations than were required because they give more information. In most (maybe all, but we're playing safe) cases, you can get full marks with rather less material than the answers given here. That is not to say that you necessarily DID get more marks with less material; it does depend on the material.

Material in Helvetica type was added after marking the test.

I occasionally correct spelling if I think it might help. These corrections have nothing to do with the marks; I do not assess spelling, punctuation, grammar, or other literary devices, even though I think that they're important. I point out mistakes occasionally because no one else is likely to tell you, so if I don't you might never find out.

While on the subject :

"receive" is spelt like that;
"extension" is spelt like that (DON'T rely on the *New Zealand Herald*, which got it wrong on 9th September);
"privilege" is spelt like that (it's a PRIVate LEGal right, not a PRIVate LEDGE; I can't think of a way to remember the second I unless you think that privilege is VILE).

TERMINOLOGY : I don't require you to get all the terminology exactly right, because there is no "exactly right" terminology in most cases, and even if there were it would probably have changed by next year. Even if you use uncommon names, I try to work out whether you have a sensible idea underneath the vocabulary. But I can't do magic, and if your terminology is wildly different from that which we've used in the notes misunderstanding is inevitable. Therefore, if you want to use terms which are different from those which appear in the notes, it is very helpful if you define them briefly before you start your answer.

PLEASE ANSWER *MY* QUESTIONS : As usual, a number of people gave (sometimes excellent) answers to questions which I hadn't asked. Examples : searching the file system in question 1, security checks in question 2, capabilities and access control lists in question 3. Much as I would like to give you credit for the knowledge displayed in these answers, I can't.

Many people answered the questions as though they were about whatever operating systems they were accustomed to using. It is (almost - just in case) certainly true that they are not. We go to some lengths in 34? to avoid being too specific; sometimes we take examples from a particular system, but we are much more concerned with general principles. If you answer a question about operating systems as though it was about (for example) Unix, it suggests that you don't in fact understand what you're talking about. Beware !

Using the same sequence of operations as the basis for all the questions was an experiment. I had hoped that being able to concentrate on just one problem would help you to form a more complete picture of it, so that you'd have a better all-round understanding and that would help you to give better answers. I am far from convinced that it worked. The necessarily simple basic problem was a constraint on setting the questions, and the result was an uneven set of questions of varying difficulty. I probably won't try it again.

QUESTION 1.

I have sometimes been accused of setting assignments which have no bearing whatever on the tests or examinations. This time, I set a question (this one) which was very similar to an assignment which you had just completed, and a significant proportion of you took no notice whatever. For example, one answer to part (ii) of case (a) was :

'It gets the signal that an icon, x, has been placed on another icon, "delete".'

(I chose that one not because it is specially good or bad but simply because it was on the first paper I marked.) As we had also spent quite a while in the lectures discussing this material, and there is a considerable amount of material in the notes on the subject, I didn't think I'd have to spell out what I wanted in any more detail than asking for "What happens at the user interface ..." and listing the specific points in (i) to (iv).

What gets the signal ? Where does the signal come from ? It certainly doesn't come from the keyboard or the mouse. It must be constructed by something from signals which *are* available, and the only something in a position to do the job is the operating system. This is a course on operating systems.

When I identify parts of questions as (i) to (iv) (or whatever), it is helpful if you identify the corresponding parts of your answers similarly. If you don't, then I suppose that you're unable to distinguish between the parts, which doesn't improve your mark. It also takes me more time; as there are around 170 people in the 34? course, every additional minute I spend per person is about three hours more work. I accept, reasonably happily, that it's part of the job, but don't much like having to spend time disentangling parts of your answers if you could have done it much more easily.

Also, if you do get as far as labelling your answers (i) to (iv), please make them correspond to the questions with the same labels. I took care to separate (ii) and (iii) in the interests of clarity, but several people mixed them together in ways which were not clear at all.

Several people took the trouble to explain how you should first make sure that the correct icon was visible (part a) or you had the correct current directory (part b). That wasn't what I intended (as is evident from the specimen answers), but I should perhaps have made it more clear in the question. Apologies to those who were misled.

The question starts "What happens at the user interface ...". People who told me about executing programmes and doing things with the file system (etc.) didn't get many marks.

The point of the question was made, probably unintentionally, in many of the answers. They contained phrases of the pattern "the system { knows | checks | understands | finds out | decides | ... }". The question is about *how* the system does all those things, which is reasonable enough in an operating systems test.

(a) What happens at the user interface when Step 3 is performed using the Macintosh-like interface :

(i) : the actions taken by the user;	(ii) : the signals which reach the system;	(iii) : what the system does with the signals;	(iv) : how the system identifies the action to be taken and the name of the file.
Move the mouse so that the pointer is on the X icon.	A stream of x and y movement signals.	On each signal, the current pointer position is recalculated accordingly; On each signal, the pointer is deleted from the screen and redrawn in its new position.	
Press the button.	A "button-down" signal	Determines (from the UIMS screen map) what is on the screen at the position of the pointer.	The X icon is found. This determines the name of the file.
Move the mouse so that the pointer is on the "delete" icon.	A stream of x and y movement signals.	On each signal, the current pointer position is recalculated accordingly; On each signal, the pointer (or icon outline, etc.) is deleted from the screen and redrawn in its new position.	
Release the button.	A "button-up" signal	Determines (from the UIMS screen map) what is on the screen at the position of the pointer.	It's the "delete" icon - therefore, as the pointer was dragging a file, the required operation is "delete".

What I particularly wanted in this answer was the requirement for a screen map (or equivalent) to interpret the raw input signals. I didn't worry too much about the other details, provided that they made sense and answered the question asked.

Whatever the details of the design, there are three steps in the path from the click to information about the object concerned : click to window object; window object to location of information about object; location of information to information.

Many answers weren't what I wanted, but could charitably be interpreted as having some merit. Because of that, they got some marks - and because of the marking scheme, they ended up with rather more than I'd intended.

Please remember that nothing happens by magic. If you say "the icon becomes shaded", you should (in this question, anyway) explain how that happens.

Answers in terms of Windows-ish interfaces (click a name in the file manager window; press delete button, etc.) were just as acceptable; the appropriate answers are quite similar to those for the Macintosh, and require much the same operations.

I accepted answers which included the "empty the trash" step for the Macintosh. Whether that is part of deleting the file is a matter of judgment; I'd rather regard it as a limited form of "undo" for the delete operation, which I think is complete once the "drag to trash" has been carried out. I point out that the delete operation to be used is defined in the question, and *does not* include any "empty trash" part.

(i) Quite a number of answers repeated the question - for example, "the user drags the icon representing the file named 'X' on to the icon which depicts a trashcan ...". I usually gave most of the marks; that was not originally my intention, but I marked from my answer sheet, and noticed that the answer given was really much like the answer on the sheet. When I noticed that it was even more like the question, it was too late to go back and change the marks.

(ii) There was very commonly a suggestion that the system receives pointer coordinate values from somewhere. Where ? The mouse doesn't calculate them (and indeed it can't), so some system component must.

I didn't worry too much about the details of the mouse signals, but there must be something to report the button movements and there must be something to report the mouse movements (and it can't be the position).

Several others said that the system received a signal "indicating that a file icon has been selected", or something similar. That's an extraordinarily complicated signal to get from nowhere in particular. Didn't some part of the system make it ?

(iii) Rather more answers than was comfortable used phrases like "the system puts the file in the trash can" or "the system drops the icon in its place" as though trash can and icon were real things. Just to make it clear : they are not real; they are artefacts of the operating system, and in this course at least you should bear it in mind.

(iv) Just knowing something happened around the "delete" icon (widely called "trash", which I accepted, but didn't say in the question) isn't enough. It has to be a button-up action when the previous actions were button-down over a file icon (or other source of a filename) and drag.

(b) What happens at the user interface when Step 3 is performed using the Unix-like interface :

(i) : the actions taken by the user;	(ii) : the signals which reach the system;	(iii) : what the system does with the signals;	(iv) : how the system identifies the action to be taken and the name of the file.
Type DELETE X	A stream of character signals.	On each signal, the character is stored in a buffer, and the current buffer pointer position is changed accordingly; On each signal, the character is displayed on the screen (and the cursor redrawn in its new position, but that's usually a hardware function).	
Type <ret>.	A <ret> character signal	Begin to determine what is in the buffer.	
		Find the first word in the buffer.	Assume it's an instruction.
		Find the rest of the string in the buffer.	Assume it's an argument needed by the instruction.

I didn't worry too much about whether or not the system "understands" the meaning of "delete"; Unix typically relies on the position in the input text string.

Many answers included bits about the system looking up the name in a table, etc. These bits were almost always irrelevant.

(i) Despite the question, a significant number of people said that the user would enter "rm X". I wondered a bit, but gave the marks anyway.

(ii) Magic again : characters entered in step (i) were quite often mysteriously transformed into signals identifying the instruction and file in step (ii).

(iv) There is no law of nature which says that the name of the operation to be performed has to be the first word in the instruction line.

QUESTION 2.

There were two parts of the question for each case : "Explain how the files ..." and "Identify the information ...". Not many people gave a clear answer to the second part, which was perhaps partly my fault for phrasing the question how I did. I tried to work out whether you knew from whatever answer you gave, but that took time.

Some answers did include lists of information required, but then gave no explanation. That wasn't much better. And notice that the information required by the system isn't just (for example) a file name; it has to know the system tables in order to use the file name.

I got quite a lot of information about access rights - despite stating that you should "ignore all matters connected with protection and security".

Many marks were lost in both parts by explanations with gaps.

In my first draft of the test paper, some parts of this question were quite similar to some parts of question 1. The overlap was not great, but was non-zero. I recognised this by including a comment that said something like "Two parts of questions are essentially identical. If your answer to either of them is 'the same as part <...>', I'll accept it and give you the same mark for both". On reflection, I decided that was asking for trouble, so cut it out and reframed the question somewhat.

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

This description is an approximation to an object-based system's view. In practice, there are frequently deviations from the ideal, so I accepted variants if they made sense.

It was not always clear whether people could distinguish between different parts of the system, because in many cases they were all referred to as "it". I would have liked to know whether you had distinguished the UIMS from the file management system, but often couldn't tell. Well, that was my fault, because I didn't ask, but if you know it does no harm to say - if nothing else, it gives the marker confidence if other doubtful points appear.

It was sometimes all too clear that people hadn't distinguished between (for example) files and icons. Quite a number inspected the icon's information to find the file's creator; even more went straight from the screen coordinates to the file.

1 : The position is identified on the **system's screen map** as lying within a window of some sort (the icon); the identity of the software associated with the window is available from the **attributes of the window**.

If you start from the screen coordinates, which is what the question says, you can't omit this step.

2 : The UIMS sends a double-click signal to the software associated with the window, which in this case is part of the file management system.

3 : The file management system identifies the file as X. (This is possible in several ways; each icon window can have its own associated software, in which case identification is immediate, or the file management system can record the positions of all icons which are visible, and search.)

4 : The double-click is interpreted by the file management system as an "open" instruction, so it inspects the **file attributes** (such as resources) of X to find its creator, P.

Some people wanted the system to work out the programme to run from the file name's extension. That's all very well in principle, but what's the extension in "X" ?

I gave some credit for the file name extension if the system extension table was mentioned.

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

1 : The first lexical item (up to the first space) of the text in the buffer (P) is identified by the shell. The Unix convention defines this as the name of the operation which must be executed.

2 : The name is compared with a **list of names** of operations which the shell can deal with; it will not be found.

3 : A file called P is sought in the **subdirectory** identified as the first to be searched in the **user's search profile**. If it is not found, the next subdirectory in the search profile is inspected, and so on. When the name is located in a subdirectory, the file P has been found.

4 : P is started, and given the rest of the text; P will then use system procedures (part of the API) to seek the file X following the same procedure as was used for P.

I didn't approve of answers which suggested that the system did something with X directly. Only P knows what to do with X - in Unix itself, only P knows what the parameters mean. This is

probably universal practice, because if you don't do it that way you have to add stuff to system tables every time you want to add a new programme to the system.

QUESTION 3.

(a) When are the protection checks performed ?

When the files P and X are opened.

That's all. Easy marks. Too easy, unfortunately - many people clearly felt that I wanted more than that, and gave me more detail. I thought I was doing you a good turn, but it didn't always work.

Some of the longer answers gave me information which you would rather I hadn't known. One example was the not uncommon sequential description which said, paraphrased, "First P is checked, then X" - demonstrating that you were already prejudging the order of events. I ignored that one, because I'd catch it in part (c) and I didn't want to mark down the same mistake twice.

The question says "when". If you didn't say when, you might not have any marks. (In one or two cases I found something in the answer which half convinced me that you know what you were doing, but that didn't often happen.)

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

The identity of the current user :	from the session information, where it was placed during the login sequence.
The identities of the files' owners :	from the file attributes.
The protection levels set for the files :	from the file attributes.

Many people (probably most ? - I didn't count) missed out the identities of the owners.

That includes people who just wrote "file attributes", without saying which. If the owners weren't specifically mentioned, I didn't count them. I'm sorry if that was wrong, but how could I tell ?

Several answers included references to "user privileges". In the system described, these are determined by the owner or non-owner status of the user; while some systems do take account of additional factors, this isn't one of them.

(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 ?

A few people - surprisingly few, in hindsight, even allowing for the test paper rubric - asked, during the test, what was meant by "both systems". That was a very reasonable question, because two systems are never defined. "A system" and "the system" appear, but neither is helpful. In fact, it was intended to mean "Unix-like and Macintosh-like systems", but it doesn't say so anywhere on the test paper. It was a careless slip on my part, and I apologise. In fact, though, it shouldn't have made any difference, because to determine which messages should be displayed you have to consider the order of access to the files anyway, and you'd just (if you answered the questions in order, which most people seem to have done) been thinking about that in question 2.

A much more serious omission, not my fault this time, was many people's neglecting to consider the combinations of the access states of P and X. Even though I specifically asked for all possible combinations, many answers were simply lists of what happened for different states of P and X independently. These answers got at most half marks.

Execute access to P	Read access to X	Message for Macintosh	Message for Unix
Forbidden	Forbidden	You may not read X	You may not execute P
Forbidden	Allowed	You may not execute P	You may not execute P
Allowed	Forbidden	You may not read X	You may not read X
Allowed	Allowed	no message	no message

Permissions for execute access to X and read access to P are irrelevant to the answer. It is just possible that a system might give different messages for attempts to execute P depending on the state of the read access permission, so I accepted answers suggesting that - but it's unlikely, as the general rule in any sort of protection or security error that might be an attempt to gain unauthorised access to something is to give as little information as possible. One answer is that all messages should be limited to "permission denied", or an equivalent. That's unnecessarily unhelpful, as you can usually work out where the difficulty is by trying to apply P to one of your own files. Notice that the question is about protection rather than security.

Some people gave me a more complicated answer including comments about read-only access. While the principle is good, the question only defined read and execute privileges, so that part wasn't required.

There was a lot of agonising over questions like whether non-owners had read access rights under certain conditions. I had carefully eliminated those by asking what happened for different combinations of forbidden and allowed access, not how to work out the access permissions. I'd also eliminated worries about read-only by not defining write privileges anyway.

There seems to be a strong impression that owners are somehow exempt from protection checks. That might be true on some systems, but it isn't universal, and there's not a lot of point in defining read and execute privileges for owners if that's the case.

QUESTION 4.

(a) A primitive system with no explicit memory management, but with upper limit and lower limit addresses provided;

The answers required here were very simple, and many people gave more than required. I didn't deduct marks for unnecessary detail, but I did require the simple answer to be there somewhere, and it wasn't always. An unsatisfactory consequence of the simplicity was that it was hard to distinguish people who really did know that the answer was small and simple from those who got it right because they couldn't think of anything else to write. I am fairly sure that there were examples of both these classes, but couldn't give more credit to those who did know what they were talking about.

The upper and lower limits give the extent of the available memory, not the size of the programme. If you want to reset them, they should still show the available memory.

In Step 1 : Apart from setting values for the upper and lower limits, the programme loader is likely to do very little. It should ensure that the upper limit is not passed when P is read into memory. (The programme must then make sure that it does not use memory outside the limits set by the system, which it can determine by convention or API call, but that isn't required by the question.)

(*Alternative answer :* The programme loader will reset the limits to show how much memory is used by the programme.)

Perhaps people were put off by the lack of material in the answer, but a number of innovative suggestions appeared. People wanted the system to read the data file into memory, but that's certainly the programme's job because the system doesn't know what the programme wants to do with the data. Virtual memory was often mentioned, but it isn't very realistic in the primitive system described.

In Step 2 : Nothing need be done.

(*Alternative answer :* The programme dismantler will return the limits to their original values.)

It was suggested that at the end of the operation the memory contents would be written back to the disc; for the programme code, there's no point, as it won't have changed, and for anything else it's none of the system's business.

Several people said that the memory was deallocated. They didn't say how; if they hadn't mentioned any sort of allocation, that didn't make much sense.

(The alternative answer gets some marks because it sounds plausible, but not all. It is not a sensible policy for such a system, because in the absence of a significant memory management system there is no point in taking any action which might restrict the memory available to any programme.)

(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 which releases a block given its initial address.

It does say "segmented". Why did several answers describe paged systems ?

A significant proportion of people misunderstood the question, and told me little or nothing about how the memory manager worked. I don't understand why; I asked you to "describe how memory is allocated ... and released ...", and that seemed to me to be a request for the mechanism. Robert wasn't surprised by my specimen answer either. Sufficient people understood it my way to show that it wasn't completely incomprehensible. Any comments ?

In Step 1 : The programme loader will request a segment of memory sufficiently large to contain the programme (or perhaps two or three segments, depending on the extent to which code, stack, and data areas are separately managed). Assuming that space is available, the segments will be allocated in the system's free space, and the memory manager will record the allocations in its memory map. If the pointer(s) returned by the memory manager is (are) not zero, the programme loader can then set up P in memory, reading the programme into the space available using the pointer returned by the API call, and setting up P's segment table accordingly. (If several memory areas are used, it must also link them together, probably by placing appropriate pointers in the segment table.)

I didn't say virtual memory, but it was not unusually mentioned in the answers. I gave a few marks for a reasonable account of how the system worked, but often the description fell well short of describing *how* the memory was managed.

Neither did I say multiprogramming, but it was often assumed. Segmenting (or paging, for that matter) is just a way of organising the internal memory. It happens to be useful for other memory operations as well, but that's not part of the segmentation. (Your television set is useful for replaying material from your video recorder, but the recorder isn't an essential part of the set.)

A surprising number of answers mentioned one of the memory map (usually, and sensibly, implemented as a linked list) and segment table, but not both.

In Step 2 : The programme dismantler must be told where the segment table is. It then works through the table, returning the memory segment by segment to the memory manager, which must record the areas as free in its memory map.

The answer to this subpart naturally depended on the answer to the previous bit, but that wasn't always obvious. Sometimes details appeared here which hadn't been given earlier.

(NOTE : I've used the non-committal terms "programme loader" and "programme dismantler" to identify system components which are obviously necessary. No further identification is required.)