Computer Science 340

Operating Systems

1998 EXAMINATION : ANSWERS AND NOTES.

Material in this font was added after marking the scripts.

GENERAL.

Some of the difficulties I found when marking my part of the paper could be resolved if, while answering the questions, you would bear in mind two assumptions about examiners :

1 : Examiners are brilliantly intelligent. They will see through your attempts to fudge answers, to cut corners, to omit parts of answers, and to change your argument in mid-question. And any other clever tricks you might try. So don't do it.

2 : Examiners are incorrigibly stupid. If you hint in one part of your answer that you know something, they will not necessarily use the hint as your answer to a different part, or another question. (They are more likely to wonder whether you understood it in the first place.) Therefore, if in a question you are asked to (for example) "Identify the information you use ...", it is advisable to include in your answer the words "Information I used :", followed by the list of things you want the examiner to notice.

QUESTION 1.

(a) The file must have two components :

1 : The file attributes, which describe various properties of the file;
2 : The file data, where the contents of the file are stored.

The operating system must manage the attributes, so that it knows the properties of the file when it needs them to do things to the file. It need not know about the contents of the file, but it must manage the space where the contents are stored, in order to control disc use effectively.

To find the file, its name is sought in some sort of table maintained by the operating system and usually stored on the file volume with the files. From other information associated with the name in the file table it is possible to find the position on the disc of each component of the file. (Commonly the file attributes are stored in the table, together with the information needed to find the file data, but other sorts of file organisation are sometimes used.)

Usually satisfactory. I did require that answers included a specific mention or implication that the location of the file is available from the file table; not everyone provided it. Some people didn't mention the file data.

File information blocks kept appearing with depressing frequency. They have nothing whatever to do with this question, except just possibly in part (c).

Several people seem to think that the file's data don't need to be managed by the system.

It would have been nice if a few more people had made the link between the attributes and the stuff in the file directory. And it's odd that while many people gave examples of file attributes comparatively few included the disc location.

(b) A Macintosh file has a third component called the resource fork. This is available for programmes to read and write in the same way as that ordinary data component, and contains various items used by the system to handle the file, but not properly part of the file data - icons are an example.

The additional "fork" might be useful if there are properties of the file which do not form part of the ordinary file data, and which would therefore be inconvenient to manage if they were kept in the ordinary data part of the file.

I did expect that the answer should include some hint on why the material was better stored separately than in the ordinary data component.

Despite the specific note in the question, many answers told me what the Macintosh did. It doesn't matter what the Macintosh does; it's the principle that's important. Several people dragged in the GUI interface, which has essentially nothing to do with the file system.

(c) Several answers are possible. Here's an example.

IMPLEMENTATION :

A file is represented as several essentially independent data arrays ("strands") stored on the disc. Each of these is accessible from the file's entry in the file table, and is managed much like a conventional file. All strands are subject to the same protection constraints, and are opened and closed together; apart from that, they operate independently.

FILE DESCRIPTOR :

The file descriptor must contain (as well as the usual items) the number of strands, and disc addressing information for each.

DIFFERENCES :

File descriptors will not be of the same length, so file tables will be harder to search - but, as the number of strands is available, it will be easy to adapt the directory search methods.

It will obviously be necessary to keep N versions of information in many cases, including N structures resembling conventional file information blocks when the files are open.

API :

When making a new file, it will be necessary to state how many strands are required.
In reading or writing, the required strand must be specified.

There were some different answers; if they were sensible, they got high marks, because they made it very clear that their authors understood what the question was about.. One was a very thorough description of a tree-structured file. Another suggestion used a fairly conventional descriptor with a floating array of indices for the difference component.

I wanted enough description of how the system worked to convince me that it would workd and (more important) that you knew what you were talking about. Simply mentioning "object-oriented file systems" wasn't sufficient.

Several people decided that the sizes of the data areas were fixed. I don't know why, but it usually made a mess of their answers. A few decided that nothing at all in the file could change. Please read the question.

QUESTION 2.

(a)

All right. I should have numbered the bits I wanted. It was often very hard to sort out which bit of answer went with which bit of question. I think I got most of it right, but it would be better for you if you made things like that clear.

Perhaps I should also have written "in ***each of*** the two systems" (in this part and the next) to make it clear that I wanted an answer for each system. The great majority interpreted the question the way I'd expected, but I tried to make allowances for the few who answered with a single method.

I think most marks were lost by people missing bits of the question. I asked for the five items in the table below, and often didn't get them. In general, people who simply worked through the question point by point, putting each bit of the answer in a new paragraph, did rather well; people who tried two write an essay did very badly.

	Textual	GUI
How to open	type "edit lectures/340/current/exam/340questions98"	Move to "lectures" icon, select; Move to "340" icon, select; … Move to "lectures" icon, select;
	I don't believe that it's easier to give a whole sequence of "cd …" instructions. Several people curiously thought that they had to use one cd and one edit. I gave the mark for that, because they included the whole directory path name in one instruction.	
My input	Keystrokes	Mouse movements, clicks
	Often omitted.	
Interpretation	First "word" is a programme name; then follow track through directories, knowing that / separates identifiers in the string, and try to open the last identifier as a file in the context found.	Integrate mouse movements, track on screen map, interpret click according to screen context and information held on the item clicked - is directory, or is file.
Information used	The name of the editor; The given file name.	The file name.
	Not many people gave an answer for this part.	
Information presented	None.	A list of file names at each step of the operation.
	Several people correctly mentioned error messages.	Several people didn't think the GUI gave them any information. One might wonder what they do with it.

The difference is that there is a way to express the whole file name in text, but not with a GUI interface.

Another possible answer, chosen by some people, is that you have to give the programme name in the textual system, but not in the GUI system. I think my answer fits the question better, because it has more to do with the interface styles; there's no reason why you couldn't attach a programme name to the file attributes in a text system, but I don't think there's any way to build a complete pathname at one go with a GUI.

I gave no marks for a collection of opinions where I asked for "one short sentence on the major difference"; I wanted you to work out which you thought was the most important. I gave one mark for any single opinion, and two for what I thought was a particularly relevant one - not necessarily mine.

Differences which had nothing to do with the question didn't do too well either. That you don't have to remember so many file names with a GUI system isn't really exhibited by an example in which you're told all the names you need at the beginning. Neither did superficial differences ("You can do it in fewer steps with a textual interface") when there's an obvious deeper reason (my answer above).

(b)

	Textual	GUI
How to open	Repeat : list current directory; look for relevant name; move to that directory until the file name appears. Then "edit 340questions98". (Backtrack if you have to.)	Repeat : list current directory; look for relevant name; move to that directory until the file name appears. Then open "340questions98". (Backtrack if you have to.)
	In some answers it was suggested that a better strategy was to start from the editing programme and just look at the files it showed. (See below for further comment.) That's an O(N) improvement, it seems to me, and makes no difference in principle. One answer at least was so framed as to suggest that the editor would list all its editable files across the whole system, which would be a severe hindrance in a system of any size.	
Information used	The name of the editor. How to list the contents of a directory. How to move to a new directory. What the file is about, and what the names mean.	What the file is about, and what the names mean.
	Very few people mentioned the use of the information in the file name. But if you can't use the (a) method for lack of information, and this method works, you must be getting information from somewhere. (Not everyone thought so : one suggestion was to conduct a random search.)	
Information presented	A list of file names at each step of the operation.	A list of file names at each step of the operation.

You have to understand the file names.

This is (obviously enough, I think) a user interface question; both sides of the interface are significant.

The first answer I got said "search for '340'". Well, it doesn't say so in the question, but you can't find "search" in any of the GUI menus and if you try it on the textual system it replies "'search' not recognised". More seriously, even if you think there "ought" to be a search operation, there's nothing in the question to suggest it, particularly when it says "***all you know*** is that you are looking for this year's examination questions". An answer of rather similar type included a suggestion that one should start the programme first, then look for the file from within the programme. But the most important objection is that it only works by accident; unless you happen to pick a significant string, it fails. I remark that this file, which is in fact just that named "340answers98", is actually called "e98acom.doc".

In both these cases you are using your knowledge of specific systems in a case when you don't know whether it's appropriate. The arbitrariness is clear in that some people said you could search with GUI but not text, some said you could search with text and not GUI, and some said you could search with both; this is obviously purely a matter of accident, and has nothing to do with operating system principles. You are entitled to assume that essential functions are available, but neither search nor opening files from within editors is essential. I gave marks if I could see any to give, but as the "search" didn't fit the question there were usually not many.

Final note on this topic : very few, if any, answers in which using "search" was recommended for part (b) also listed "search" as a recommendation for the GUI part of part (a). Why ?

(No, I didn't take marks off if people didn't specify an editor name with the textual system, nor if they didn't explicitly use "cd" to list the files in the textual system, provided that they were consistent.)

Many people gave answers which amounted to "enter the file name". I don't know how you can do that if you don't know the file name, and no one who did that gave any explanation.

(c) In finding the way through the directories, we use a lot of information implied by the file names.

This illustrates the importance of having file names which describe the function and nature of the objects they identify. The full path name includes many attributes of the file, and without some provision for segmenting this information, such as is provided by the directory model, one would be forced either to use exceedingly long file names or to do without the information.

This part was often done well; using the silly names must have hammered the point home !

For high marks, I expected some suggestion that the directory levels were connected with things we know about the file.

In one or two answers, there was no suggestion that we might have to understand the names. They got few marks.

A curiosity : some people answered by saying that there was no logical connection between the numerical identifiers and the nature of the files; others said that the numerical identifiers gave only logical information. It made no difference to the marks. I think the second was more correct; the information missing is semantics, not logic.

QUESTION 3.

(a) For each segment, the system maintains :

Item	*Significance*	*Purpose*
Present bit.	The segment is, or is not, currently in memory.	To determine whether a segment fault is necessary.
Memory address.	Where the segment begins in memory.	To convert virtual addresses into real addresses.
Segment length.	How long the segment is.	To check memory range errors.
Disc address.	Where the segment is, or can be put, when swapped out.	To store or retrieve the segment when needed.

(You can get by with less; you don't need two address fields if you don't mind allocating a new disc address each time the segment is swapped out. An answer with a single field is acceptable, provided that appropriate comment is given. Other items - access bits, dirty bits, etc. - are not essential. The segment number is not strictly required - if the table is big enough for all possible segments it's just an index - but I gave credit if it was included.)

For full marks I wanted at least address, length, present bit, and some mention of the disc. Curiously, many answers didn't mention the disc at all. Long lists of items suggested that their authors didn't know what they were talking about, so in extreme cases I reduced the mark for inappropriate suggestions.

(b) The working set of a process is the collection of its segments which must be in memory at a particular time if the process is to carry on running without serious hindrance from virtual memory activity.

This was in the nature of a gift if you knew the answer. You were supposed to know the answer, but not everyone did. I expected the answer to contain some suggestion that the working set was connected with limiting memory faults, and a hint that it depended on the current activity of the process.

It is NOT the set of segments currently in memory, or all the segments for the programme, o9r the number of segments that the process can use at a time.

(c) Important factors in determining the behaviour are the number and size of the segments (both determined by the value of the parameter N), the demand for memory in the system, and the pattern of access to the segments.

The pattern of access is illustrated by the diagram; each cell represents an element of the array, and the numbers in the cells show the order in which they are required in memory.

1, 2, 3, 4	5, 7	9, 11	13, 15	17, 19	21, 23
6, 8	4N+1, 4N+2 4N+3, 4N+4	4N+5, 4N+7	4N+...	4N+...	4N+...
10, 12	4N+6, 4N+8	8N+...	8N+...	8N+...	8N+...
14, 16	4N+...	8N+...	12N+...	12N+...	12N+...
18, 20	4N+...	8N+...	12N+...	16N+...	16N+...
22, 24	4N+...	8N+...	12N+...	16N+...	20N+...

It is clear that the first row of the matrix is in steady demand until the end is reached, after which there is a steady demand for the second row, and so on. The numbers down the left-hand side show that there is also another, quite different, pattern; each row in the matrix is required twice, but then ignored for a time depending on N.

For small N ($N^2 < W$), both the number of segments and their size is small. Unless memory is very congested, it is quite likely that all segments can be brought into memory at once, and that the interval between the successive references to the lower rows is sufficiently short that they will still be available when needed again. If these conditions are satisfied, the algorithm will run smoothly.

As N increases ($N^2 \approx W$), the long gaps between references to the lower rows make it likely that the rows will be swapped out before they are required again, so greatly slowing down the algorithm, and further delaying the successive references : a memory fault is likely to be generated for each row once this pattern is established. The comparatively frequent references to the current "top" row are more likely to keep it in memory.

With yet larger values of N($N^2 > W$; $N \approx W$), even the top row may be swapped out between cycles, leading to even slower operation. An additional constraint is the possible difficulty in finding a large enough memory area to accommodate the large segments.

I expected both the analysis of the algorithm and the qualitative description (including relationships between N and W) for full marks. Answers which amounted to "all right if small, thrashing if big" didn't get many marks.

QUESTION 4.

This area is complicated by the stupidity and incompetence of several software providers, who have misused words like "backup" and "archive" with meaning quite different from the originals. I warned you about that, but short of inventing yet another set of new words there's not much we can do about it. I went to some trouble to give you careful definitions of how I would use the terms in the course, and I assume that you will use the terms that way (or state carefully that you're using them in some other way) when you answer examination questions. Clearly, several of you have not troubled to do that, but that leaves me with a set of statements which are clearly wrong in my terms, and I don't know whether they're right or wrong in the unknown (to me) terms which you're using. I can't give you credit for such answers. This does not worry me a lot, because one clear implication is that you are unaware that the vocabulary is imprecise, and I think that's something youi should know.

(a)

I specifically asked for the functions of the services, not their implementations. I got quite a few implementations, with some people spending a lot of time telling me about complete and incremental backup techniques, which was just what I was trying to avoid. I also specifically asked for the advantages of the services, which I often didn't get.

System backup is a system-wide mechanism designed to provide protection from accidents, which aims to ensure that a system file store can be returned to its state as it was at some - preferably recent - known time. Its main *advantage* to the system's users is that it limits the damage which can be done by a system collapse to the work done since the last backup operation.

"Copy all the files" isn't what it does; it's how it's done.

"System backup - this is when the whole system is backed up ..." isn't a very convincing answer, particularly when the meaning of "backup" is never defined in the answer. (And it doesn't help that in the same answer there appears "File generations - this is when a backup of a file is made ...".) How am I supposed to assess your understanding from that ? (Those are quotations - but someone else gave almost the same "answers", with the second also adding "File archives - storage of many back up files ...".)

File generations are intended to offer protection against erroneous alteration or removal of files. It ensures that previous versions of files are automatically saved, so that they can be retrieved in case later versions are in any way damaged. The main *advantage* of this technique is that it provides an "undo" function for any operation which results in gross changes to a file.

A **file archive** is a long term, off-line store in which people can save files as and when they decide to do so. It provides safety against system failure, as does a backup system, but adds the *advantage* of long term storage and owner's control over storage time and contents. Backup systems only guarantee to restore the system to its state at the most recent backup; even though older versions of files are often retrievable from an incremental backup system, the total life time is not defined and backup times are not related to the times of significant changes to the files. An archive system provides facilities for better file management, and can reduce the demand for the system disc by providing storage elsewhere.

Compression is not an essential feature of an archive, though it's obviously useful.

(b)

It is curious that when I specifically excepted working details from the answer in (a) you gave them, but now when I ask you to "Describe how the system works", you don't.

How it works : From time to time, the automatic archive must search the disc directories for files to be archived, using some criterion such as time since the file was last used or amount of disc space available.

EITHER	*OR*
Any files found must be copied to the archive medium (disc or tape), and transported to the archive store.	Any files found must be copied through appropriate communications media to a distant site.

The files are then deleted from the local disc. As the archived files must appear in the system directory in the usual way, their directory entries remain, but must be labelled as representing files in archive, and provided with whatever information is needed to initiate a request to retrieve the files when necessary (typically some sort of address).

Necessary changes : As the automatic archive is in effect an extension to the conventional system, it must obey the conventional system instructions as far as possible, so that operations (copy, delete, rename, etc.) requesting actions on archived files must as far as possible be amended to work correctly, applying deletions, changes of name, etc. to the archived files.

EITHER	*OR*
Attempts to perform operations impossible from archive - such as opening the file - must be reported in appropriate error or warning messages.	Attempts to perform operations which cannot be handled locally must be handled by communicaion with the remote system, possibly by fetching the file concerned to the local disc and deleting from the distant site.

Not many people suggested any "necessary changes".

You also need a "last-used date" for each file, which must be provided if not already present. I didn't include it in the answer because many systems provide something of the sort, but accepted it if offered.

Several people remembered the diagram from the notes, but in most cases didn't give any evidence of knowing what it was about.

Many people worried a lot about how to start up the archiver each day, and gave me lots of details. In practice, many systems provide all manner of clock and calendar facilities for setting things off at predefined times, and as it isn't in the file system ut has nothing to do with the question anyway.

(c) (This was a real question. When it was first set, the Computer Science Department had just begun to operate such a backup discipline, but without an index.)

EITHER	*OR*
Yes.	No -
The function of the file archive is to provide a long term off-line store for selected files; this function would be satisfied by the eternal backup if the "selected" requirement could be met. It is not met automatically, as, though the backup saves all the files, it only does so at certain times, so one can no longer decide to save copy of a file in the archive at arbitrary times.	provided that one is willing to replace the "archive X" operation by "copy X under a new name which I'll remember somehow to my directory called ARCHIVE and delete it from there automatically after a month". In other words, one can make do, but at the cost of some inconvenience. This is exploiting the save-forever feature but managing without the immediate effect.

Notice that the answer follows from the function of the archive, not from its implementation. Suggestions that it isn't an archive because it isn't implemented like one are beside the point; if it can be made to provide the same service, then it's a possibility.

For full marks, I expected some discussion of the position of files changed since the last full backup. I very rarely got it.

I received a strong impression that several of the answers could be restated as "I have no idea what to do, but if I waffle a bit about reliability or convenience or cost there might be a mark or two". In most cases, it didn't work.

A few people gave answers equivalent to "Yes, if we do not have a proper indexing facility for the ... backup". That's a good point, and I gave some marks for it.

QUESTION 5.

(a) A traditional process consists of resource requirements (memory, open files, security domain etc) and one stream of execution (register values, PC, stack pointer and stack).
A thread is a stream of execution. Therefore multiple threads can be part of the same resource environment (usually referred to as a task or process). They share the same environment including memory.
Switching context from one thread to another in the same resource environment is simpler than switching context from one traditional process to another. This is because far less work needs to be done. There is no need to swap page tables as the threads are accessing the same memory, other tables such as open file tables also remain the same. The only things which need to be changed are thread dependent registers (including access to the thread stack).

(b) The threads and processes are similar in that the they share the same code and the same open files at the time of creation (thread or process).
The major difference is that the forked processes have different data over time. File tables are originally the same but may change as the processes run. Whereas with three threads everything stays shared and changes to memory by one thread are visible to all. The forked processes would have to communicate via some external means if they wanted to communicate (unless they had a shared segment of memory, but I don't expect many to think of that).

(c) If the system knows about threads it can handle them properly. In particular a thread which executes a system call and blocks will not stop other threads running. In a user level thread system the operating system only dispatches the process, the process itself dispatches one thread at a time. If that thread is blocked in a system call, all threads in that process are blocked.
A related advantage is that the operating system can allocate a time slice to each thread rather than to the process. Design decisions have to be made here as to whether a process with 10 threads should get 10 times the processor time of a process with a single thread. Under the user level thread system this decision cannot be made.

QUESTION 6.

(a) Location transparency - there is no connection between the name or identifier of a resource and its physical location on the system.
Migration transparency - resources can be moved around the system without processes which use the resource being aware of the change. A slightly weaker form of this is referred to a location independence in the text.
Replication transparency - resources may be replicated throughout the system. Processes which use a replicated resource are not aware of the multiple copies.

(b) The rmiregistry program must be running.
The remote Java machine constructs an object which implements the remote service.
This remote service object is registered, along with a name for the service, with the rmiregistry.
The client queries the rmiregistry for an object which is associated with the service name.
The rmiregistry responds with a remote reference to the object. The client's stub object then uses the remote reference for all RMIs.

(c) There has to be an intermediate remote object running on the host specified in the URL which passes the requests on to the host with the "real" remote object. The intermediate object must also implement the remote interface. The local client actually gets a reference to the intermediate object. Care would have to be taken to ensure that the intermediate object passed all method calls including inherited ones such as equals() on to the "real" remote object.

QUESTION 7.

(a) B: 0
A: 1
B: 2
A: 3 ...

(b) "i" must be incremented by three, which means this procedure will stop until two other advances are given (by the other two procedures). The main program would have to be:

```
OneAtATime(A, 0);
OneAtATime(B, 1);
OneAtATime(C, 2);
```

(c) Event counters don't require mutual exclusion to implement them. In some cases such as the example above only one event counter is required to synchronise two concurrent processes. The equivalent would require two semaphores.

Processes using semaphores to coordinate activity don't need to keep hold of a separate count themselves whereas they do with event counters. This makes it difficult to use event counters for general resource allocation. In the example in the question each procedure call has to have a different starting value for "i".

QUESTION 8.

(a) Device table -

an operating system structure which contains a device descriptor for each device in the system; the system's source of information on its devices.

Device descriptor -

a structure which contains information about a single device, providing or pointing to all the information needed to use the device. This includes such things as the state of the device, what it can do, routines for standard operations, location of buffer areas, and so on.

Input-Output request block -

an IORB is a structure representing a single request for service from a device. This may be a request for information, or a control instruction, or an actual data transfer.

Device driver -

a device driver is code which handles the administration of a device at the hardware level.

Interrupt handler -

device-specific software which receives interrupts from its device. Different types of interrupt are distinguished, and dealt with appropriately.

(b) Smoothly to withdraw a device from service and install a replacement, it is necessary to ensure that transactions intended for the old version are satisfactorily completed, software changed to that needed to manage the new device, and communication with the system reestablished for normal running.

Closing down the old device :

It must be possible to mark the device as temporarily unavailable in the device table, so that new attempts to open the device can be detected and handled by the system. (What the system does in such a case depends on circumstances, and isn't part of the question, but it may be possible to use an alternative equivalent device (if there is one), or to queue the request until the changeover is complete (sensible for a batch system), or engage in dialogue with someone wishing to use the device interactively.) Transactions already in progress can be completed; if the change isn't urgent, processes which have begun to use the device may be permitted to continue until they close it.

Removing the old device :

The device may require special closing-down operations; these can only sensibly be incorporated into the device driver. Also, as well as physically unplugging the device and carting it away, it must be removed from the system tables. Its device table entry must be removed, and its interrupt handler (if any) must be unlinked from the interrupt handling system and deleted. Finally, the device driver (and interrupt handler) must be deleted.

Installing the new device :

This operation is the inverse of the previous one. The new interrupt handler and device driver must be loaded into the system, and linked into the input-output system as appropriate.

Making the new device available :

If this device is to be used as a full replacement for the old one, its device table entry must in some respect contain identical information. This is typically some sort of device type field, set by operator instruction when the device is installed, and interrogated by the file system when a request to open a file using a specific device is received. With this field set, the device can be made available, and all should proceed properly.