COMP25111

Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Operating Systems

Date: Friday 25th January 2013

Time: 14:00 – 16:00

Please answer Question ONE and any TWO Questions of your choice from the other THREE questions provided

For full marks your answers should be concise as well as accurate. Marks will be awarded for reasoning and method as well as being correct

---

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]

1. **Compulsory**

a) Explain what the term ‘deadlock’ means, and how it might occur.

(2 marks)

Multiple processes are in execution, but each is waiting for some other process to wake it up, and this is never going to happen because all the other processes are similarly waiting. Deadlock typically occurs because synchronisation between the participating processes is somehow flawed. This can be by design (because the problem to be solved is essentially insoluble) or by accident (because the programmer made a mistake).

b) Computers typically operate in one of two modes, namely *user mode* or *system mode* (the latter is sometimes called privileged mode, supervisor mode or kernel mode). Explain why these two modes are needed and how they differ from each other.

(2 marks)

1b.

For security, some things can only be done by the OS (e.g. changing page table entries, dealing with interrupts, swapping processes, etc.) and hardware enforcement is necessary (otherwise nothing can stop the user from doing whatever they want). So system mode adds some functionality which the user cannot get at directly (only indirectly, by making a system call). Typical examples of additional functionality are: access to certain parts of memory and certain parts of hardware (peripheral registers, etc.), special instructions, and so on.

c) Briefly explain how *starvation* may occur in process scheduling.

(2 marks)

At least one process gets continuously overlooked by the scheduler. A typical scenario is where new incoming processes are systematically placed at the head of a single scheduling queue, thereby repeatedly deferring later entries. Other scenarios are possible

d) Why is the size of the time slice in pre-emptive scheduling algorithms chosen to be significantly higher than the time taken for a context switch?

(2 marks)

A context switch is essentially 'wasted' time – no process benefits from it. Hence the number of context switches needs to be minimised, which in general can only be achieved by making them less frequent, subject to other necessary constraints. It is easy to see that a time slice of the same size as a context switch will lead to only 50% utilisation of the CPU for processes, so in practice we need a time slice that is one or two orders of magnitude larger (one order larger implies around 90% CPU utilisation – two orders implies around 99%).

e) Explain what is meant by the term ‘multiprogramming’.

(2 marks)

1.e

Bookwork (2 marks):

The following points should be covered to some degree in the answer:

Answer should cover the ‘basic points’ and for higher marks put these basic points into context.

Basic points:
Multiprogramming has been used in the past.
Multiprogramming loads multiple programs into the physical memory.
When using Multiprogramming the operating system switches between the multiple programs and switches them into the physical memory.
When one program has finished [the Multiprogramming operating system] bring in a new program.

Points contextualised explicitly:
In a multiprogrammed system, it is necessary to have more than one program in the memory at one time.
This is because it would be grossly inefficient to save and load processes to/from disk every time a process switch occurs.
As it is ,in general, impossible to know which combination of processes will be needed in memory, it would be very restrictive if a program had to be loaded at the same set of addresses every time.
Therefore, a scheme is needed where a program can be loaded starting at any convenient memory location, this is relocation.

2 marks for an answer that describes and contextualises multiprogramming and gives the salient facts in a sensible way; and provides a well-defined set of brief descriptions of each;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lectures 10 Memory Management (1).

TOTAL marks (2 marks) [2]

f) Explain the difference between a virtual memory address and a physical memory address.

(2 marks)

1.f

Critique [Differentiate] (2 marks):

The following points should be covered to some degree in the answer:

A physical memory address is an address which corresponds to a fixed location in physical RAM memory.
A virtual address is an address in a conceptual memory space which a process (program) thinks it is operating.
Virtual addresses are normally translated to physical addresses by Memory Management Unit (MMU) hardware.

2 marks for an answer that depicts all the salient facts in a sensible way; and correctly delineated and briefly described;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lecture 11: Virtual Memory (1).

TOTAL marks (2 marks) [4]

g) What is a 'page replacement algorithm'?

(2 marks)

1.g
Book (2 marks),
Example answer:- The following points should be covered to some degree in the answer:

In a virtual memory, the memory space is divided into pages. There are normally more pages than can fit into the real (RAM) memory of the system and thus some pages are held on background storage to be moved into main memory only when the CPU wants to access them. When a page needs to be moved into main memory, it is necessary to decide which page to reject to make space for it. The page replacement algorithm is used to make the decision of which page to reject.

2 marks for a totally correct explicit delineation of a page table using keywords in context.
1 mark for some basic table (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 13; Virtual Memory (3); Introduction to: Page Replacement Policies.

TOTAL marks (2 marks) [6]

h) What is meant by ‘memory mapped I/O’?

(2 marks)

1.h
Bookwork (2 marks),
Example answer:- The following points should be covered to some degree in the answer:

There are two basic ways that a CPU can communicate with a peripheral device. Early CPUs used special I/O instructions which included specific peripheral numbers to select data and control information from a device. Modern CPUs use normal memory load and store instructions to read/write from/to areas in the memory address space. This is memory mapped I/O.

2 marks for a totally correct explicit delineation of a page table using keywords in context.
1 mark for some basic table (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 14; Controlling Input and Output (1).

TOTAL marks (2 marks) [8]

i) What is a 'page fault'? Describe how a page fault is handled by the Memory Management Unit and the operating system.

(2 marks)

1.i
Bookwork (2 marks),
Example answer:- The following points should be covered to some degree in the answer:

In a paged virtual memory system, pages may have copies in real memory or may exist only on backing storage (usually disk or secondary memory). If the MMU attempts an address translation but the page "**table indicates that there is no copy in real memory, this is a page fault. The MMU interrupts the CPU and traps to an OS page fault handler. This is responsible for both selecting a page to reject from real memory – and writing it to disk if it has changed since it was last read" – and** organising the read of the required page from disk to real memory. It then updates the page tables and transfers control back to the original program to re-execute the instruction which caused the fault.

2 marks for a totally correct using keywords in context,
1 mark for some basic table (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 11, Virtual Memory (1), Paged Virtual Memory.

TOTAL marks (2 marks) [10]

j) Briefly explain why Java code that waits in a synchronized method for a condition to hold will commonly retest the condition when it has been released from its wait.

(2 marks)

Typically the waiting thread will not be alone, and it is often then more convenient to wake up all waiting threads which will then have to ensure that they can proceed 'safely'. They do this by re-testing the condition that they have been waiting for, and resuming their wait if it is still not true.

2. a) Explain briefly the difference between a process and a thread.

(2 marks)

A process is a complete executing program, including its virtual memory space as defined by page tables and whatever else is needed. A thread is an execution image living in (and sharing) some existing process' virtual memory space. The 'private' memory attached to this is much smaller than the entire virtual memory space (effectively a stack and the CPU state).

b) Explain briefly what a CPU burst is and what an I/O burst is. What is a CPU-bound process, and what is an I/O-bound process? Why is it a good strategy in process scheduling to give higher priority to I/O-bound processes?

(4 marks)

A CPU burst is where a process uses the CPU continuously for some (relatively long) period of time. An I/O burst is where a process uses the CPU intermittently while it is mostly waiting for data transfer to occur between CPU and peripheral devices. A CPU-bound process is mostly executing in CPU bursts, while an I/O-bound process is mostly executing in I/O bursts. I/O-bound processes voluntarily yield the CPU more readily than CPU-bound processes, thereby ensuring that "someone else gets a go" more often. Progress towards the final goal of completing execution is thus quicker (and there are other benefits, such as reduced contention for resources).

c) Three processes A, B and C all alternate between a CPU burst of 3 time-units and an I/O burst of 3 time-units. Draw a diagram showing the states of these processes as they are run by a Round Robin scheduler for a total of 30 time-units, assuming that they all start ready at time-unit 0, the time-slice adopted by the scheduler is 2 time-units, and the time for a context switch is negligible. For what percentage of the time is the CPU executing user processes?

(4 marks)

Diagram should look something like the following (=== is I/O burst, -- is "ready but not running"):

```
AA----A===AA----A===AA----A===
--BB---B===-BB---B===-BB---B==
----CC--C=====CC--C=====CC--C=
          *          *          *
```

CPU is idle at *s above, so utilisation for user processes is 90%.

d) A new scheduler is introduced using priority queues. There are two queues, Q1 (responsible for scheduling processes A and B) and Q2 (responsible for scheduling process C). Assuming they have ready processes, the two queues access the CPU alternately, as follows: Q1 gets 5 time-units, then Q2 gets 2 time-units, then Q1 gets 5 time-units, and so on. Processes in each queue are executed in Round Robin fashion with a time-slice of 1 time-unit. If a queue runs out of ready processes before its allocated time-units have been used, it yields access to the CPU to the other queue. Apart from this, the situation is as described for part c) above. Draw a diagram showing the states of the three processes A, B and C as they are run by the scheduler for a total of 30 time-units. For what percentage of the time is the CPU executing user processes?

(4 marks)

Diagram should now look like the following:

```
A-A-A===AAA===A-AA===A-A--A===
-B-B---B===B-B-B===-B-B-B===BB
-----CC-----C=====CC-----C===-
              +               *
```

Note: Q2 gives up the CPU at + above because it has no "ready" process to run.
CPU is idle at * above, so utilisation for user processes is about 97%.

e) In a certain system, the execution of three threads is synchronised using three semaphores, S1, S2 and S3, as shown below. Semaphores S1 and S2 are initialised to zero, while semaphore S3 is initialised to 1. All three semaphores are used only in the sections of code shown below.

Thread A	Thread B	Thread C
…	…	…
P(S1)	P(S2)	P(S3)
P(S1)	P(S1)	V(S1)
x=3*x+4	x=x+7	x=x*5
V(S2)	V(S2)	V(S1)
V(S1)	V(S1)	V(S3)
…	…	…

i) If the variable x is defined as an integer shared variable, initialised to 1, and is not assigned a value in any other sections of the code apart from those shown above, what will be its value when all threads have finished executing? What will be the values in the three semaphores? Justify your answers.

(3 marks)

Threads A and B block until S1 and S2, respectively, are vacated. This can only be done at this stage by thread C, which vacates S1 first and thereby potentially releases thread A. Thread A now tries another procure for S1 which can again only be vacated by thread C. This happens only after thread C has executed its update of x, so this is the first action (x=5*1=5). Now thread A can be released, and it does its update of x. Nothing can happen in thread B, and thread C does not touch x again, so this is the second action (x=3*5+4=19). Now thread A vacates S2, which releases thread B which now tries to procure S1. This is possible once thread A has vacated S1. Then and only then thread B can do its update of x, so this is the third action (x=19+7=26). It is also the final update to x since no more are left, so the final value of x is 26. Thread B then vacates S2 and S1, and there is nothing to stop thread C vacating S3, so all semaphores will be vacated at the end (S1=S2=S3=1).

ii) Would it be possible to guarantee the same final value for `x` using only two semaphores? Would it be possible to guarantee the same final value for `x` using only one semaphore? Justify your answers.

(3 marks)

S3 plays no part in synchronisation between the threads, and we are told that no other code uses the semaphores, so it is possible to remove the associated P and V operations. Thus the answer to the first question is that it is possible to guarantee the same final value for x using only two semaphores. We can even do so when some of the P and V operations are removed. But removal of another semaphore is not possible because we need to ensure **both** that thread A is not released until thread C has done its update **and** that thread B is not released until thread A has done its update. If only one semaphore is used for both tasks, it will not be possible to guarantee to hold up both A and B at the start of the codes and this could lead to early updating by thread B.

3. a) On a paged machine with 3 pages frames (PFs) available for it, a particular process makes accesses to the following pages in the order given:

0, 3, 7, 1, 3, 2, 1, 3, 7

Show the contents of the 3 page frames and the cumulative total number of page faults (PF) after each memory access assuming that an LRU page replacement algorithm is in use and that the page frames are initially empty. The type of diagram you should draw up is depicted in Figure 3a.

(4 marks)

Access:	0	3	7	1	3	2	1	3	7		
Most recent:	X	X	X	X	X	X	X	X	X	)	Contents of
Second most :	X	X	X	X	X	X	X	X	X	)	page frames
Third most:	X	X	X	X	X	X	X	X	X	)	in memory
Total PFs:	X	X	X	X	X	X	X	X	X		

Figure 3a. Typical diagram showing 3 page frames and the cumulative total number of page faults.

3.a
Application (4 marks),
Example answer:- The following points should be covered to some degree in the answer:

Access:	0	3	7	1	3	2	1	3	7		
Most recent:	0	3	7	1	3	2	1	3	7	)	Contents of
Second most :	-	0	3	7	1	3	2	1	3	)	page frames
Third most:	-	-	0	3	7	1	3	2	1	)	in memory
Total PFs:	1	2	3	4	4	5	5	5	6		

4 marks for a totally correct content, 3 marks for all 3 page frames totally correct and 1 mark for the cumulative total number of page faults totally correct.
2 marks for half issues covered,
1 mark for some basic calculations (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 13; Virtual Memory (3).

TOTAL marks (4 marks) [4]

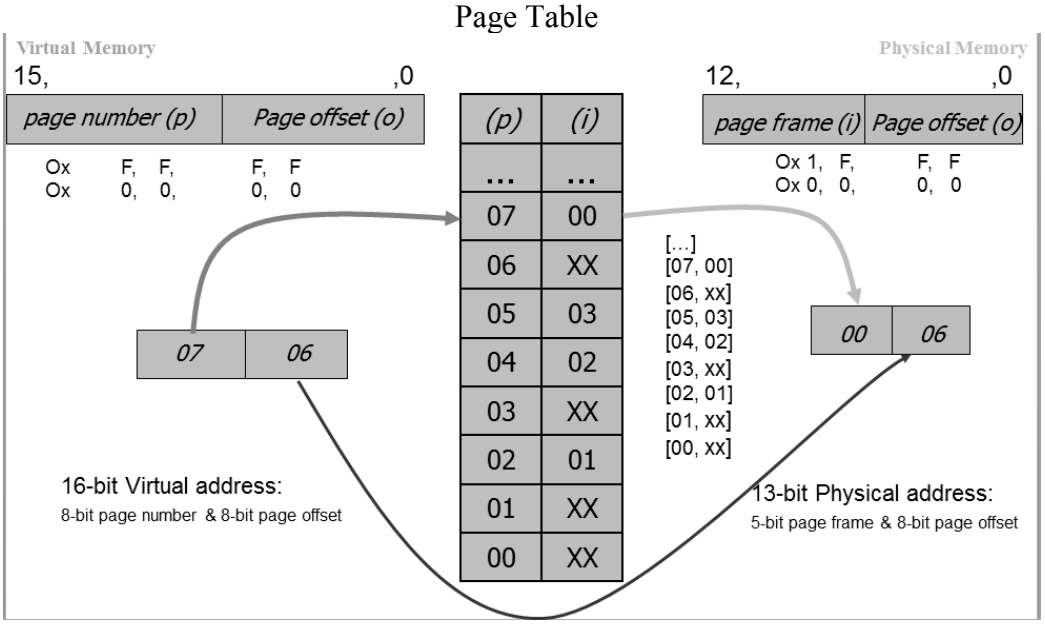
b) A 16-bit virtual address is divided into an 8-bit page number and an 8-bit page offset; whereas the associated physical address has a 5-bit page frame with the appropriate page offset address sizing. Show, with the aid of a fully labelled diagram, the table structure necessary to convert this virtual address into a real physical address.

(6 marks)

3.b

Application (6 marks),

Example answer:- The following points should be covered to some degree in the answer:



6 marks for a totally correct content, 2 marks virtual memory totally correct, 2 marks for example page table totally correct and 2 mark for physical memory totally correct. (totally correct: implies sizing of page no & page off. as well as sizing of page frame & page off.)

3 marks for half issues covered,

1 mark for some basic calculations (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 11; Virtual Memory (1).

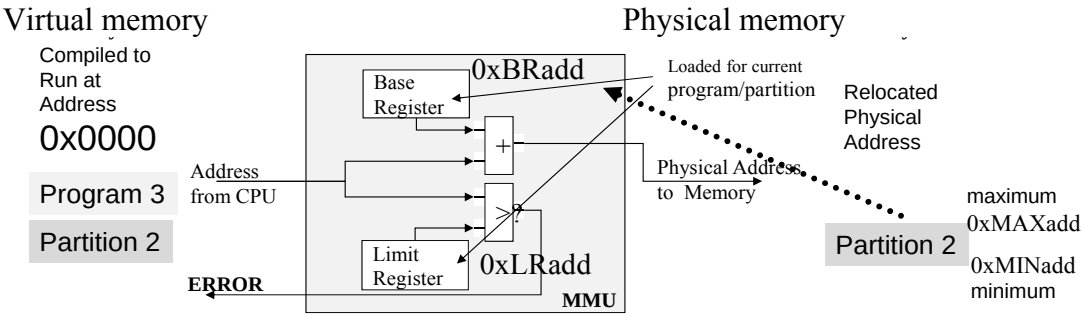
TOTAL marks (6 marks) [10]

c) With the aid of a diagram, describe the structure of a simple base and limit system and explain how it achieves relocation. **NOTE**: For full marks your answer must contain a detailed concise diagram and explanation; e.g. a fully labelled diagram and a concise explicit description of a base and limits system and how it achieves relocation.

(6 marks)

3.c
Bookwork (6 marks),
Example answer:- The following points should be covered to some degree in the answer:

A base+limit system is a piece of hardware which is placed between the CPU and memory. It has a 'base address' register which can be loaded as execution of a program is scheduled and this is added to any address issued by the processor. Note, if the program is represented in binary as if it starts at address 0, this will automatically address the program as if it starts at the base address. (Standard diagram required).
The limits register compares the address issued by the processor to the maximum partition address; if greater it throws an error, setting the ERROR line.



Question Answer figure 3.c. Base and limits diagram.
Note: MMU implies memory management unit; '+' implies addition; '>' implies compare [if greater than throw ERROR].

6 marks for a totally correct [fully labelled] diagram and, explicit delineation of a base and limits system using keywords in context
3 mark for some basic calculations (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 10; Memory Management (1).

TOTAL marks (4 marks) [16]

d) Given a 1G address space and associated 128K page size, calculate the number of pages in the virtual address space. **NOTE**: To gain full marks you must show full working.

(2 marks)

3.d
Application (2 marks),
Example answer:- The following points should be covered to some degree in the answer:

Virtual address space 1GB and page size 128KB.
If the virtual address space is 1GB and the page size is 128KB there are:

$$\text{Number of Pages} = \frac{\text{Address space}}{\text{Page size}}$$

$$\frac{1\,GB}{128\,KB} = \frac{2^{30}}{2^{17}} = \frac{1{,}073{,}741{,}824}{131{,}072} = 2^{30-17} = 2^{13}\,(or = 8{,}192)$$ = 8K pages

2 marks for an answer that calculates the correct answer and is laid out correctly e.g.
2 marks for a correct answer and full working out,
1 mark for a 'right lines' approach. Moderate marks will be awarded in the case of correct application for a wrongly calculated.,
½ **marks** for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 11; Virtual Memory (1).

TOTAL marks (2 marks) [18]

e) Given a physical address space of 2G and associated 64K block size, calculate the number of page frames in the physical address space. **NOTE**: To gain full marks you must show full working.

(2 marks)

3.e
Application (2 marks),
Example answer:- The following points should be covered to some degree in the answer:

If the virtual address space is 2GB and the block [page] size is 64KB, there are:

$$\text{Number of Page frames} = \frac{\text{Address space}}{\text{Block size}}$$

$$\frac{2G}{64K} = \frac{2^{31}}{2^{16}} = \frac{2{,}147{,}483{,}648}{65{,}536} = 2^{31-16} = 2^{15} (or = 32{,}768) = 32K \text{ page frames}$$

2 marks for an answer that calculates the correct answer and is laid out correctly e.g.
2 marks for a correct answer and full working out,
1 mark for a 'right lines' approach (moderate marks will be awarded in the case of correct application wrongly calculated),
½ **mark** for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 11; Virtual Memory (1).

TOTAL marks (2 marks) [20]

4. a) Explain how a peripheral communicates with the CPU using interrupts.
(3 marks)

4.a
Bookwork (3 marks):
The following points should be covered to some degree in the answer:

A peripheral wanting to make a transfer raises a hardware signal [interrupt line]. The CPU completes its current instruction and then enters a routine to service the peripheral. This may either be by a single routine address (in which case the routine must use polling to identify the peripheral) or a device specific address via a table. The CPU then reads from or writes to the device before resuming the original program that was interrupted. The address required to do this is saved by the hardware when the interrupt occurs.
Other keywords that may be utilised [in context]: interrupt acknowledgement – IACK, processor saves the current value, interrupt vector, Interrupt Service Routine (ISR)..

3 marks for an answer that depicts all the salient facts in a sensible way; and the use of keywords in context;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lectures 15: Controlling Input and Output 2.

TOTAL marks (3 marks) [3]

b) Segmented memory is an alternative view to that adopted by paged memory; the following questions relate to segmented memory.

i) Explain the difference between a *page* and a *segment* in a virtual memory system.

(3 marks)

4.b.i

Critique [Differentiate] (3 marks):

The following points should be covered to some degree in the answer:

Very basic answer is: Page is fixed size, segment is variable size.
In a paged virtual memory system, the pages are all the same size.
In a segmented virtual memory system, the segments are of variable size.
Segmentation is less about mapping a larger virtual address space onto a smaller physical memory (the purpose of paging), and more about supporting the operating system in general.

2 marks for an answer that depicts all points, and the use of keywords in context;
1 marks for half of the facts;
½ mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lectures 12: Virtual Memory (2), Segmented Virtual Memory.

TOTAL marks (3 marks)[6] Final mark RNs is [10]

ii) A computer system uses segmented virtual memory only (no pages). The state of the memory at a given time is shown in Figure 4b. Indicate what happens when a new segment requiring 9KB of memory space is loaded using the following algorithms:

1) Best Fit; and
2) First Fit.

State where the 9KB will be placed given 1) and 2), and, due to the different algorithms, if any issues arise. Note that it is assumed that the lowest address is at the bottom of the diagram, and the third column indexes all the segments A, …, J.

(4 marks)

	9 KB Gap	J
	15 KB Segment	I
	8 KB Gap	H
	20 KB Segment	G
	10 KB Gap	F
	10 KB Segment	E
	8 KB Gap	D
	18 KB Segment	C
	14 KB Gap	B
Lowest address is at the bottom of the diagram →	9 KB Segment	A

Figure 4b. Typical diagram showing segmented memory.

4.b.ii

Application (4 marks):

The following points should be covered to some degree in the answer:

1) The Best Fit algorithm will map it to the 9KB gap [index J] at the top of the memory

2) The first fit algorithm will place it in the 14KB [index B] gap. This will leave an additional 5KB gap between the new segment and the 18KB [index C] segment.

4 marks for an answer that depicts all points, and the mentioning of the 5KB gap;
2 marks for half of the facts;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lectures 12: Virtual Memory (2), Segmented Virtual Memory.

TOTAL marks (4 marks) [10]

c) A *pathname* in a hierarchical file system defines how a file is to be located on the disk.

i) Explain the distinction between an *absolute pathname* and a *relative pathname*.

(2 marks)

ii) Describe in detail the algorithm used to locate a file given an *absolute pathname* or given a *relative pathname*.

(3 marks)

iii) In a system using *index nodes* (*i-nodes*), the above algorithm will first find the identifier of the i-node that describes the target file. Explain how the identified i-node is found on disk.

(2 marks)

iv) Assume that each i-node contains up to 8 pointers to the first 8 successive blocks of the file on disk, then up to one pointer to a *single indirect block* on disk that contains 512 further pointers to the next 512 successive blocks of the file on disk, and then up to one pointer to a *double indirect block* that contains 512 further pointers to single indirect blocks on disk that each contain 512 pointers to the next 512 successive blocks of the file on disk. Any unused pointers contain a special 'null' value. What is the size (in blocks) of the largest possible file?

(3 marks)

Bookwork and Application (10 marks),
Example answer:-
The following points should be covered to some degree in the answer:

4c(i) Pathname is a string of text. An absolute pathname starts with separator and defines path starting at the root directory. A relative pathname starts with a filename and defines path starting at the current/working directory.

2 marks for a totally correct answer,
1 mark for partially correct answer (or attempt).

4c(ii) Starting from appropriate directory, pathname defines alternatively a directory (a special kind of file) name then a separator then a directory name then a separator (and so on) until (finally) a filename. Algorithm follows each directory specified until either it finds no corresponding directory or file, or it finds the desired file. For each directory on the path, it must (open, if necessary, and then) read the corresponding (directory) file until it finds the appropriate subdirectory or (finally) file.

3 marks for a complete and correct answer,
1 mark deducted for each (serious) defect.

4c(iii) The i-nodes are stored in a special area of the pertinent disk partition. Each i-node has identical size. The i-node identifier is an appropriately sized index into the special area of disk.

2 marks for a totally correct answer,
1 mark for a partially correct answer (or attempt).

4c(iv) 8 blocks can be accessed directly, plus 512 for the single indirection, plus 512×512 for the double indirection, giving a possible maximum size of (262,144 + 512 + 8) = 262,664 blocks.

3 marks for a totally correct calculation,
1 mark deducted for each (serious) error.

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 16, 18.

TOTAL marks (10 marks) [20]

END OF EXAMINATION