Université d'Ottawa
Faculté de genie

École d'ingénierie et
de technologie de l'information

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

CSI 3310
Operating System Principles

FINAL EXAMINATION - SOLUTIONS

Length of Examination: 3 hours **April 27, 2005, 9:30**

Professor: Stefan Dobrev

Family Name: _____

Other Names: _____

Student Number: _____

Signature _____

Closed book.

If you do not understand a question, clearly state an assumption and proceed.

No calculators or other electronic devices are allowed.

At the end of the exam, when time is up:
- Stop working and turn your exam upside down.
- Remain silent.
- Do not move or speak until all exams have been picked up, and a TA or the Professor gives the go-ahead to leave.

The exam consists of three (3) parts

Part 1	**Multiple Choice**	**12 points**
Part 2	**Short Answers**	**21 points**
Part 3	**Problem Solving**	**22+2 points**
Total		**55+2 points**

50 points = 50% of the total mark

Multiple Choice Questions (12 questions, each of them for 1 point)

1) Process A may be moved from the Running state to the Wait state when:

- **a)** Another higher priority process enters the Ready state.
- **b)** The **activate** signal occurs.
- **c)** **Process A executes a system call.**
- **d)** The event it is waiting for occurs.
- **e)** The interrupt handler returns control to executing program.

2) Scheduling of user threads (in a pure user-level thread system):

- **a)** **Is the responsibility of the user program running a thread library.**
- **b)** Is the responsibility of the kernel.
- **c)** Is the responsibility of system thread scheduling process.
- **d)** Is the joint responsibility of the user program and the kernel.
- **e)** None of the above.

3) Which of the following is not an IPC (inter-process communication) mechanism?

- **a)** sockets
- **b)** **monitors**
- **c)** shared memory
- **d)** pipes
- **e)** remote procedure calls (RPC)

(actually, all of them involve IPC to some level, but for monitors it is a bit of a stretch)

4) One of the following statements about monitors is false:

- **a)** Monitors are high level synchronization constructs that can be used to control access to shared variables.
- **b)** Monitors, unlike semaphores, allow the synchronization code to be encapsulated in one place, reducing the chance of error.
- **c)** **Monitors can be entered only by calling wait() or signal() for a conditional variable in the monitor.**
- **d)** If there is no process or thread waiting for a conditional variable x, then issuing x.signal() will do nothing.
- **e)** At any moment of time only one process can be inside the monitor.

5) Only one of the following rules would prevent deadlocks:

a) A process must always simultaneously request at least half of all resources.
b) A process can request resources only if it does not hold any resources, or it holds at least half of all resources.
c) A process can request only one resource at a time.
d) A process can hold at any time only one instance of each resource.
e) All resource types are numbered 0,1,..K, where K is the last resource type, and a process can request and/or hold only the resources of the same parity (i.e. holding R1 and requesting R3 is permitted, but holding R2 and requesting R3 is forbidden)

6) One of the following statements is false about deadlocks:

a) In the banker's algorithm, if a process requests less resources then there are available, its request might still be denied.
b) If there is deadlock then there must be a process that holds one resource and waits/requests another resource.
c) If there is only one instance of each resource, deadlock can be detected by checking whether there is a cycle in the resource-allocation graph.
d) **Before a process is admitted into a system guarded by deadlock detection algorithm, it must specify how much of each resource will it ever need.** That is for banker's algorithm, for deadlock detection you assume a process would not request more.
e) If the system is in unsafe state, the deadlock is still not certain and might be avoided.

7) One of the following statements is true about fragmentation:

a) Allocation into fixed partitions incurs zero internal fragmentation, but terrible external fragmentation.
b) If paging is used, there is zero fragmentation.
c) Due to increasing page table sizes, the smaller the pages, the larger the internal fragmentation problem is.
d) **Internal fragmentation denotes the unused memory within the address space of the process.**
e) External fragmentation means that the process's address space is divided into many small fragments spread throughout the physical memory.

8) One of the following statements is true about TLB:

a) TLB stands for Time-Lapse Buffering and comes into use when DC (Don Cherry) is transmitted over CBC (Circuit-Buffer-Circuit) channel.
b) TLB allows all page table entries to be accessed directly on the CPU, without incurring memory-access penalty.
c) TLB caches frequently accessed memory locations, eliminating the need for page tables
d) A TLB miss means that page fault has occurred.
e) **TLB is an associative memory caching the frequently used page table entries..**

9) Which of the following is NOT considered a file attribute

a) File Name
b) File Size
c) **File Data Pointer**
d) File Protection bits
e) File Modification time

10) One of the following statements is true:

a) Disc cylinder consists of all tracks on the same platter.
b) **All blocks of the cylinder can be read without moving the disc head.**
c) As the inner tracks are shorter then the outer tracks, the harddisk spins faster when reading the inner tracks, to maintain *constant linear velocity*.
d) The latency and transfer rate of the harddisk is primarily determined by the I/O bus used.
e) If the hardisk's head crashes, the data can be recovered by cleaning the surface and replacing the head.

11) One of the following statements is false about kernel I/O subsystem:

a) I/O buffering is used in order to match different speeds of producing and consuming devices (including the CPU).
b) I/O buffering is used to provide copy semantics, e.g. when executing asynchronous write() or send(), to have the caller be able reuse the buffer immediately after the call returns.
c) I/O buffers are typically stored in kernel space, and cannot be paged-out.
d) The access to the hardware devices is controlled by I/O instructions being privileged instructions that can only be executing in kernel mode, and by having the memory addresses of the memory-mapped I/O ports being outside of the address space of the user processes.
e) **Device controller is a software layer controlling the hardware device by writing commands to device's I/O port and by responding to I/O interrupts raised by the device.**

12) One of the following statements is true:

a) Blocking I/O does not involve interrupt handler.
b) DMA is preferred mode of communication between the CPU and the keyboard
c) Direct I/O is used for bulk transfer of data from devices directly connected to the I/O bus.
d) **Unlike direct I/O, interrupt-driven I/O incurs context-switch overhead for each data transfer.**
e) Devices whose data rate is at least 10^5 times slower then the front-side-bus of the CPU should be avoided, as the CPU wastes too much cycles waiting for them.

Part 2 – Short Answer Questions (21 points)

Question 1. (3 points)

The following table shows how 4 physical page frames allocated to a process have been loaded with the process virtual pages. All values are numbered using decimal values and starting at 0. All timing values correspond to the number of clock ticks from the start of the process execution. The R bit is a reference bit that is set whenever the loaded page is referenced.

Virtual Page Number	Page Frame Number	Time loaded	Time Referenced	R Bit
2	0	60	161	0
1	1	130	160	1
0	2	26	162	1
3	3	20	163	1

For each page replacement algorithm below, circle the virtual page numbers that cause a page fault in the given sequence of virtual page accesses. Under the page number that causes the page fault, show the page replacement by giving the new list of loaded virtual pages (i.e. the contents from the first column of the above table). It is not necessary to show the contents of the list for the virtual pages that do not cause a page fault.

Optimal	4	0	0	0	2	4	2	1	0	3	2
0	2									2	
1	1									**3**	
2	0									0	
3	**4**									4	

(among the virtual pages 2,1,0,3, 3 is the furthest in the future. At the end, we do not have enough information about the future to decide, we just know that 2 will not be replaced)

Clock	4	0	0	0	2	4	2	1	0	3	2

Note that the pointer is set to Page Frame 2.

frames init										
0	2	4*	4*	>4*	>4*	>4*	3*	3*		
1	1*	>1*	>1*	1	1*	1*	>1	>1		
2	>0*	0	0*	0	0	0*	0	0		
3	3*	3	3	**2***	2*	2*	2	2*		

LRU	4	0	0	0	2	4	2	1	0	3	2
0	2							2		2	
1	**4**							4		**3**	
2	0							0		0	
3	3							1		1	

Question 2. (4 points)

Simulate page buffering using FIFO replacement algorithm. There are 4 physical frames allocated to the process, and two more frames are used for page buffering. Your task is to complete the following table. The Buffer List column depicts the mapping of the virtual pages into physical frames, in the following format:

<Virtual Page Number> in <Physical Page Frame> (Modified Bit)

Thus the entry "3 in 3 (1)" indicates that the virtual page 3 is loaded into the physical frame 3 and that the page has been modified. The Free and Modified Lists have entries with the same format but without the modified bit.
There are 2 pages total found in the Free List and Modified List combined. Initially the Modified list is empty and two pages are available in the Free List as shown.

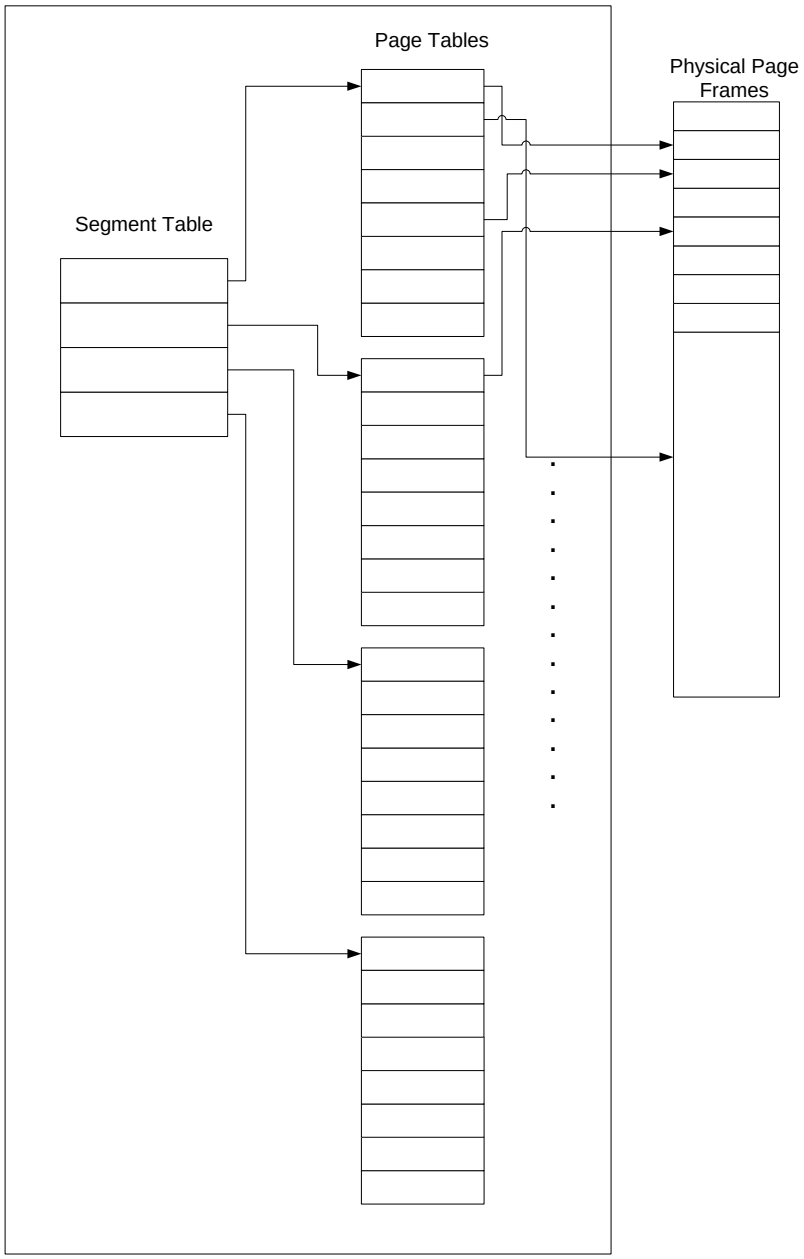
Operation	Buffer List	Free List	Modified List
Initial	3 in 3 (1) ← 0 in 2 (0) 2 in 0 (0) 1 in 1 (1)	- in 4 - in 5	
After "Read 4"	0 in 2 (0) 2 in 0 (0) 1 in 1 (1) 4 in 4(0)	- in 5	3 in 3
After "Write 0" (executed after previous operation)	0 in 2 (1) 2 in 0 (0) 1 in 1 (1) 4 in 4(0)	- in 5	3 in 3
After "Write 3" (executed after previous operation)	2 in 0 (0) 1 in 1 (1) 4 in 4(0) 3 in 3(1)	- in 5	0 in 2
After "Read 6" (executed after previous operation)	1 in 1 (1) 4 in 4(0) 3 in 3(1) 6 in 5(0)	2 in 0	0 in 2

Question 3. (4 points)

Consider the following simple memory management system combining segmentation and paging (illustrated in the figure to the right):

A process consists of at most 4 segments, where each segment itself is paged, with pages/frames of size 2kb. The page table of each segment contains 8 entries, each storing reference to the corresponding physical frame. The segment table entries refer to the page tables of the corresponding segments.

Because of relatively small size, both segment table and segment page tables are stored directly in the process's PCB (process control block), without occupying another physical frames.



Answer the following questions:

a. What is the maximum size of each segment?

   16kb (8x 2kb)

b. What is the largest logical address for the process?

   65535 = 2^16-1 (16kb x 4 segments, starting with 0)

c. Give the format of the logical address.
   xx yyy zzzzzzzzzzz

   xx – segment #
   yyy – page # within the segment
   zzzzzzzzzzz(11 bits) – offset within a page

d. Explain how is the logical address translated to the physical address.

   1. take xx and use it as an index into the segment table to get the pointer to the page table of that segment
   2. take yyy and use it as an index in this page table, get the frame #fffff
   3. concatenate the offset to the frame number to get the physical address fffffzzzzzzzzzzz

Question 4. (3 points)

The following four tables capture the contents of the page tables for processes P1, P2, P3 and P4.

Process P1	
Frame #	**Present bit**
-	0
5	1
-	0
10	1
1	1
7	1

Process P2	
Frame #	**Present bit**
-	0
-	0
2	1
-	0
-	0
8	1

Process P3	
Frame #	**Present bit**
-	0
-	0
6	1
-	0
11	1
3	1

Process P4	
Frame #	**Present bit**
-	0
4	1
12	1
-	0
-	0
14	1

Assume the physical memory contains 16 frames, give an Inverted Page Table representation for the same allocation of the physical frames to these four processes.

Your task: Fill the entries in the Inverted Page Table below:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	P1,4	P2,2	P3,5	P4,1	P1,1	P3,2	P1,5	P2,5	-	P1,4	P3,5	P4,2	-	P4,5	-

Question 5. (2 points)

Explain how resident set management policy based on page fault rate works. Under what circumstances will a whole process be suspended/swapped out?

The system monitors the page fault rate (# of page fault per fixed time slot) for each process. If the page fault rate is higher then an upper threshold, the resident set size of the process is increased, if the rate is lower then the lower threshold, the resident set size is decreased. If the resident set size cannot be increased (because all other processes also have high rate of page faults), a process is chosen and swapped out.

Question 6. (2 points)

Consider a harddisc with 32 tracks numbered 0..31 and assume that the queue of I/O requests contains requests for tracks 7, 11, 25, 4, 16 and 14 and initially the head is on track 13, moving towards higher numbered tracks.

Describe the head movement for each of the following disc scheduling algorithms:

a) FIFO: 7,11,25,4,16,14

b) SSTF: 14, 16, 11, 7, 4, 25

c) SCAN: 14,16,25,31,0,4,7,11

d) C-LOOK: 14,16,25,4,7,11

Question 7. (3 points)

What techniques are used in RAID to achieve high performance and reliability? Give two examples how are these techniques combined (explain two RAID levels of your choice that combine at least two of those techniques).

data striping (bit striping/block striping) – gain performance using parallelism
disk mirroring – reliability through duplication, fast, but wastes half of the available storage space
ECC/parity bits – reliability through redundant information, less storage space wasted, but not as fast

Examples of combining these approaches:

RAID level 5 – combine striping with (distributed) parity bits
RAID level 0+1 (or 1+0) – mirror a striped set of discs (or stripe the data over mirrored discs)

Part 3 – Long and problem solving questions (22+2 points)

Question 1 - Deadlocks (5 points)

The following system is a snapshot of 5 processes (P1 to P5) sharing 4 resources (R1 to R4).

$$\text{Max} = \begin{Bmatrix} 0 & 0 & 1 & 2 \\ 2 & 7 & 5 & 0 \\ 6 & 6 & 5 & 6 \\ 4 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \end{Bmatrix} \quad \text{Allocated} = \begin{Bmatrix} 0 & 0 & 1 & 2 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 4 \\ 2 & 3 & 5 & 4 \\ 0 & 3 & 3 & 2 \end{Bmatrix} \quad \text{Available} = \{ 2, 1, 0, 0\}$$

a. Compute the Needs matrix (i.e. show the amount of resources each process might need).

Simply Max – Allocated:

0 0 0 0
0 7 5 0
6 6 2 2
2 0 0 2
0 3 2 0

b. Show that the system is in a safe state.

1. **P1 can be satisfied, as it needs (0,0,0,0) are less the Available**
2. **After P1 finishes, it releases (0,0,1,2) to Available, which is now (2,1,1,2)**
3. **P4's need (2,0,0,2) can now be satisfied**
4. **After P4 finishes, it releases (2,3,5,4) to Available, which is now (4,4,6,6)**
5. **P5's need (0,3,2,0) can now be satisfied.**
6. **After P5 finished, it releases (0,3,3,2) to Available, which is now (4,7,9,8)**
7. **P2's need (0,7,5,0) can now be satisfied**
8. **After P2 finished, it releases (2,0,0,0) to Available, which is now (6,7,9,8)**
9. **P3's need can now be satisfied – all processes have been satisfied**

c. If a request from P3 for the resources (0, 1, 0, 0) arrives, can it be granted immediately? Show how you arrived at your answer

Assume that request has been granted, we get:
Allocated to P3: (0,1,3,4), Avaliable: (2,0,0,0)

1. **P1 can be satisfied, as it needs (0,0,0,0) are less the Available**
2. **After P1 finishes, it releases (0,0,1,2) to Available, which is now (2,0,1,2)**
3. **P4's need (2,0,0,2) can now be satisfied**
4. **After P4 finishes, it releases (2,3,5,4) to Available, which is now (4,3,6,6)**
5. **P5's need (0,3,2,0) can now be satisfied.**
6. **After P5 finished, it releases (0,3,3,2) to Available, which is now (4,6,9,8)**
7. **Neither P3 nor P2 can now be satisfied, the request should not be granted.**

Question 2 - Semaphores (5+2 points)

Using semaphores simulate a unisex washroom using the following rules:

1. At most 3 people can be in the washroom.
2. Men and women cannot be in the washroom at the same time.
3. useWashroom() is the method that simulates a person using the washroom.
4. Complete the methods maleUse() and femaleUse().
5. Define the semaphores and variables you are using, including initialization values and a short description.
6. Your solution must not deadlock, but do not worry about starvation.
7. You will get additional +2 points for a solution without starvation (i.e. steady stream of one sex does not block the other sex from entering the washroom).

Hints: *(You do not have to do it this way, but the following hints might help you come up with a solution that actually works.)* Use variables `maleNum` and `femaleNum` to denote the number of males/females that requesting the washroom (either waiting for it, or already inside). You want semaphores `maleAccess` and `femaleAccess` to control access to these variables. In addition, you want semaphores `maleEnter` and `femaleEnter` for controlling the number of males/females entering the washroom. Finally, a semaphore `empty` might be useful for gaining access to the washroom when the washroom is empty.

Solution: P166-169 in the Downey's book on semaphores

Semaphores:

Variables:

```
maleUse()
{
```

```
        useWashroom();

}
```

```
femaleUse()
{

      useWashroom();

}
```

Question 3 – Hierarchical Page Tables (4 points)

Consider a ``computer'' with 9-bit logical address space using hierarchical paging with pages of size 8 bytes. The lowest 3 bits of the logical address are used to specify the offset within the page, the highest 3 bits are used to specify the page of the page table and the middle 3 bits specify the entry in the given page of the page table. The highest bit of a page table entry is interpreted as valid/invalid bit, the remaining bits reference frame # where the corresponding logical page is stored.

Assume the content of the master page table and the page table blocks is as follows:

Master Page Table (stored in frame 0):
0000000, **1**001000, **1**010010, **0**000000, **0**000000, **1**101110, **0**000000, **0**000000, **0**000000

Pages 0,3,4,6,7,8 of the page table: <not in memory>
Page 1 of the page table: in frame 001000 = 8, contents:
0000000, **1**110111, **1**011010, **0**000000, **0**000000, **0**000000, **0**000000, **1**001111
Page 2 of the page table: in frame 010010 = 18, contents:
1110000, **1**110001, **0**000000, **0**000000, **0**000000, **0**000000, **1**000100, **0**000000
Page 5 of the page table: in frame 101110 = 46, contents:
1010000, **1**010001, **1**010010, **1**010011, **1**010100, **1**010101, **1**010110, **1**010111

A) **(2 points)** Describe the actions performed by the memory management system when the process executes instruction LOAD R1, 010010010 (load into register R1 the content of the logical memory with address 010010010). In particular:

- **a.** list which entries of the master page table and which pages and entries of the page table are examined by the system in order to compute the physical address
  - i. look at entry 010 = 2 in the MPT :101000, it is present, in physical frame 001000=8
  - ii. look at entry 010 = 2 of the Page 2 of the page table: 0000000, not present, page fault
- **b.** determine whether there are page faults and if yes then how many
  - i. yes, one
- **c.** if there are page faults, assume the pages are loaded into frames starting from 100000
  - i. the page is loaded into frame starting at 100000
- **d.** determine the physical address referenced
  - i. so the physical address is 100000010

B) **(2 points)** Assume the next instruction is LOAD R2, 100011110. Again, describe the actions performed by the MM system.

- **a.** Look at entry 100 = 4 in the MPT
- **b.** Page fault, load page 4 of page table into frame 100010 and set entry 4 of the MPT to 1100010
- **c.** Look at entry 011 =3 in page 4 of the page table
- **d.** Page fault, load the page into frame 100011 and set entry 3 of the page 4 of the page table to 1100011
- **e.** The physical address is 100011110

Question 4 - File Management (8 points)

Consider the following simplified file management system:

- the disk has 32 blocks
- each block contains 8 bytes/items
- block 0 is the boot block
- block 1 contains the root directory
- block 2 contains the bitmap of the used blocks (only the first 4 bytes are used, the most significant bit of the first byte corresponds to the block 0, the least significant bit of the fourth byte corresponds to block 31).
- each directory block contains 4 entries in format (name, index block), if the name begins with capital letter, it means a subdirectory
- remaining blocks contain files/subdirectories or are empty

Assume the following content of the disc (the 8-character strings describe the content of the file (and empty) blocks, the content of the index/directory blocks is shown in a human-readable format):

What	**Block #**	**Content**
Boot block	0	(booting code)
Root directory	1	(abba: 14 , bubba: 8 , Xtra: 6 , dubba: 22)
Bitmap of free blocks	2	00011100000001000000010001100000
<empty>	3	162345dg
<empty>	4	00000000
<empty>	5	9fh3li9f
Directory *Xtra*	6	(Ytra: 7 , ceta: 9 , <empty> ,<empty>)
Directory *Ytra*	7	(zeta: 11 , <empty>,<empty>,<empty>)
Index block of file *bubba*	8	10,12,15,16,17,-1,-1,-1
Index block of file *ceta*	9	23,24,18,19,-1,-1,-1,-1
Block 0 of file *bubba*	10	Abhkjdls
Index block of file *zeta*	11	31,30,29,-1,-1,-1,-1,-1
Block 1 of file *bubba*	12	26423642
<empty>	13	2lh9k3jj
Index block of file *abba*	14	20,-1,-1,-1,-1,-1,-1,-1
Block 2 of file *bubba*	15	1642wf32
Block 3 of file *bubba*	16	2fty432d
Block 4 of file *bubba*	17	2r5yw434
Block 2 of file *ceta*	18	917r4829
Block 3 of file *ceta*	19	2r302933
Block 0 of file *abba*	20	20fj3830
<empty>	21	kf7h4kd9
Index block of file *dubba*	22	27, 28,-1,-1,-1,-1,-1,-1
Block 0 of file *ceta*	23	3g5u4323
Block 1 of file *ceta*	24	2fy5qeg5
<empty>	25	6kfh38f4
<empty>	26	00000000
Block 0 of file *dubba*	27	ajd8d3dd
Block 1 of file *dubba*	28	dks83jdd
Block 2 of file *zeta*	29	skass9dj
Block 1 of file *zeta*	30	0293kjd8
Block 0 of file *zeta*	31	33d33d33

:

A) **(2 points)** Fill the 'What' column in the table on the previous page, using the following terms

- Empty block
- Index block of file ____
- Directory ___
- Block #___ of file ____

(you can use abbreviation to save on typing)

B) **(1 point)** Assume initially disc came formatted with 0s everywhere. Explain how it can happen that there is an empty block whose content is different from 00000000.

The file that formally contained that block has been deleted.

C) Consider the following code snippet:

```
fd = open("Xtra/ceta", "r");
read(fd, buf, 18);
close(fd)
```

(1 point) What is the content of the buffer `buf`?

Simply the first 18 bytes of file ceta: 3g5u43232fy5qeg591

(2 points) List the blocks accessed by the filesystem (assume no caching) during the execution of this code. List them in the order they were accessed.

1. **#1 (root directory)**
2. **#6 (directory Xtra)**
3. **#9 (index block of ceta)**
4. **#23,24,18 (data blocks of ceta)**

D) **(2 points)** Assume that the same files and directories, placed in the same blocks are represented in FAT, which is in blocks 2,3,4,5. Note that you don't need the index blocks and the bitmap anymore, but you still need the directories and the directory entries have now different content.

Your task: Fill the numbers in the ____ fields in the following table:

What	**Block #**	**Content**
Boot block	0	(booting code)
Root directory	1	(abba: 20_, bubba: _10_ , Xtra: 6 , dubba: _27)
FAT block #0	2	_-1,__-1_-1,_-1,_-1,_-1,__-1,_-1_
FAT block #1	3	_0_,__0,_12,_0_,_15,_0_,_0_,_16
FAT block #2	4	17_,_-1,_19,_-1_,_-1,_0_,_0_,_24
FAT block #3	5	18_,_0_,__0,_28,_-1,_-1_,_29,_30_
Directory *Xtra*	6	(Ytra: 7 , ceta: _23 , <empty> ,<empty>)
Directory Ytra	7	(zeta: 31 , <empty>,<empty>,<empty>)