



%Exam : Operating Systems 1 (OS 1)% (Solution)

Duration: 1h 30 - Documents: not allowed	First Name :
Academic year : 2025 / 2026	Last Name :
Level : L2 Semester : 5	Group :
Date : 10/05/2025 (08 : 30 – 10 : 30)	Score : ··· / 20

Exercise 1 : (Understanding Questions) (4.5 Points / 15 Minutes)

Q1) What are the two components of a virtual address used in segmented virtual memory?

Segment Number (#) and offset (0.25 + 0.25 = 0.5 pts)

Q2) Identify the following in one phrase or sentence (do more than just expand the acronym, though).

- a) **PTE (Page Table Entry) :** PTE (Page Table Entry), used to map virtual to physical addresses, stores access and protection bits. (0.5 pts)
- b) **Working set :** In a given time interval, the set of pages required to prevent page faults; used in the Working Set model for page replacement. (0.5 pts)
- c) **Thrashing :** A condition where the system spends most of the time transferring pages to and from the disk and little time making progress on application computations; usually due to memory being over-subscribed or a poor page replacement algorithm. (0.5 pts)

Q3) The text section of a process contains the program code; what are the other sections of the memory allocation for a process?

The **stack, heap and data** sections. (0.25 + 0.25 + 0.25 = 0.75 pts)

Q4) The CPU scheduling can be preemptive and non-preemptive, what is the main difference between them?

A **non-preemptive** scheduling is evoked only when the current process running on the CPU gives up the CPU **voluntarily either due to the termination** of the process or the completion of its current CPU burst (for example waiting for I/O). (0.5 pts)

Otherwise, **preemptive** scheduling allows a process to be **interrupted in the midst of its execution**, taking the CPU away and allocating it to another process. (0.5 Pts)

Q5) Would increasing the number of frames always decrease the number of page faults for a particular reference string for FIFO? Why or why not?

No. (0.25 + 0.5 = 0.75 pts)

Belady's anomaly states that it is possible to have more resources, and worse access behavior.

Exercise 2 : (Logical Addresses & Physical Addresses) (3 Points / 15 Minutes)

Q1) In paging, given the logical address **0xAFF9** (in hexadecimal) with a page size of **512 bytes**, what is the page number and offset in decimal?

A 1010	F 1111	F 1111	9 1001
$2^n = 512 \Rightarrow n = 9$			
101 0111		1 1111 1001	
Page # = 87 (0.5 pts)		Offset = 505 (0.5 pts)	

Consider the following segment table (in hexadecimal) :

Segment #	Limit	Base
0	0x1000	0x1400
1	0x0400	0x6300
2	0x0400	0x4300
3	0x1100	0x3200
4	0x1000	0x4700

Q2) What are the physical addresses (in decimal) for the following logical addresses (segment # and offset)?

a) < 0, 4 548 >

(0x0100) = 4 096

4 548 > 4 096, so, Illegal reference (error) (0.5 pts)

b) < 4, 1 048 >

0x1000 = 4 096 & 0x4700 = 18 176

1 048 < 4 352, so, Physical @ = 18 176 + 1 048 = 19 224 (0.5 pts)

Q3) What are the logical addresses (segment # and offset) for the following physical addresses (in decimal)?

a) 16 394

We have : 0x3200 = 12 800 and 0x1100 = 4 352

12 800 < 16 394 < 12 800 + 4 352 => 12 800 < 16 394 < 17 152

=> segment # = 3

Offset = 16 394 - 12 800 = 3 594

Logical @ = < 3, 3 594 > (0.5 pts)

b) 6 154

We have : 0x1400 = 5 120 and 0x1000 = 4 096

5 120 < 6 154 < 5 120 + 4 096 => 5 120 < 6 154 < 9 216

=> segment # = 0

Offset = 6 154 - 5 120 = 1034

Logical @ = < 0, 1034 > (0.5 pts)

Exercise 3 : (Unix Process Creation) (2.5 Points / 10 Minutes)

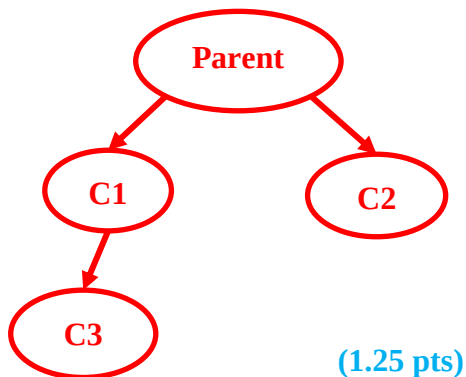
Consider the following section of the code and assume that all fork system calls are successful.

```
void main() {
    int j = 2;
    pid_t pid;
    while (j >= 1) {
        pid = fork();
        j=j-1;
        if (pid > 0) {
            printf("In parent %d. \n", j);
        }
        else {
            printf("In child %d. \n", j);
        }
    }
}
```

Q1) What value is returned if **fork** system call is not successful?

-1 (0.5 pts)

Q2) Draw a process tree for the above program.



Q3) What will be printed?

In parent 1. (0.25 pts)

In parent 0.

In child 1. (0.25 pts)

In parent 0.

In child 0. (0.25 pts)

In child 0.

Exercise 4 : (CPU Scheduling) (4.5 Points / 20 Minutes)

Consider a single-processor system and the four processes **A**, **B**, **C** and **D**, which perform computation and I/O with a disk according to the times given below.

Arrival Time	Process	1 st CPU	1 st I/O	2 nd CPU	2 nd I/O	3 rd CPU
1	A	2	3	2	3	2
3	B	4	1	4	-	-
4	C	2	1	2	-	-
7	D	1	1	1	1	1

The processes are available from the start, in this order : **A** then **B** then **C** then **D**. Give the execution charts for the following scheduling algorithms:

Q1) CPU scheduling policy is FCFS (First Come First Served) (Assume FCFS for I/O) (1.25 pts)

CPU		A	A	B	B	B	B	C	C	A	A	D	B	B	B	B	C	C	A	A	D		D			
QUEUE					C	C	C	A	A	D	D	B	C	C	C	C	A	A	D	D						
							A	D	D	B	B	C			A	A	D	D								
									B		C					D										
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

I/O				A	A	A		B		C		A	A	A	D							D				
QUEUE													D	D												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Q2) CPU scheduling policy is Round Robin (RR) with Quantum=2 (Assume FCFS for I/O) (1.25 pts)

CPU		A	A	B	B	C	C	B	B	A	A	D	C	C	B	B	A	A	D	B	B	D				
QUEUE					C	B	B	A	A	D	D	C	B	B	A	A	D	D	B		D					
							A	D	D	C	C	B				D	B	B								
									C		B															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

I/O				A	A	A		C		B		A	A	A	D					D						
QUEUE													D	D												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Q3) Calculate the Turnaround Time (TR) and the Waiting Time (WT) for each process. (1 pt)

	Turnaround Time (TR)				Average	Waiting Time (WT)				Average
	A	B	C	D		A	B	C	D	
FCFS	19	13	14	16	15,5	13	5	10	13	10,25
RR with Quantum=2	17	18	10	15	15	11	10	6	12	9,75

Exercise 5 : (Paging) (3 Points / 15 Minutes)

Consider a system using paging technology and having the following characteristics:

- A page table with **2^{16} entries**
- Each page table entry is coded on **16 bits**. An entry contains a frame number reference and a presence/absence bit.
- The offset is coded on **16 bits**
- A virtual address is indexed by **2 Byte**

Answer the following questions, always justifying your answer:

Q1) What is the size of a page?

$$\text{Page Size} = 2^{(\text{offset})} \times \text{index virtual @} \quad (0.25 \text{ pt})$$

$$\text{Page Size} = 2^{16} \times 2 = 2^{17} \text{ Byte} = 128 \text{ KB} \quad (0.25 \text{ pts})$$

Q2) How many frames does physical memory contain?

$$\text{Size of a PT entry} = \# \text{ number of bits to code a frame} + 1 \text{ presence/absence bit}$$

$$\# \text{ number of bits to code a frame} = \text{Size of a PT entry} - 1 \text{ presence/absence bit} \quad (0.25 \text{ pts})$$

$$\# \text{ number of bits to code a frame} = 16 - 1 = 15 \text{ bits}$$

$$\# \text{ of frames} = 2^{\# \text{ number of bits to code a frame}} = 2^{15} \text{ Frames} \quad (0.25 \text{ pts})$$

Q3) What is the size of physical memory?

$$\text{Size of Physical memory} = \# \text{ of frames} \times \text{frame size} \quad (0.25 \text{ pts})$$

$$\text{We have : } \text{frame size} = \text{page size} = 128 \text{ KB} = 2^{17} \text{ Byte}$$

$$\text{Size of Physical memory} = 2^{15} \times 2^{17} = 2^{32} = 4 \text{ GB} \quad (0.5 \text{ pts})$$

Q4) What is the size of virtual memory?

$$\text{Size of virtual memory} = \# \text{ of pages} \times \text{page size} \quad (0.25 \text{ pts})$$

$$\text{We have : } \# \text{ of pages} = \# \text{ of entries in PT} = 2^{16} \text{ pages}$$

$$\text{Size of virtual memory} = 2^{16} \times 2^{17} = 2^{33} = 8 \text{ GB} \quad (0.5 \text{ pts})$$

Q5) What is the size of the address bus in bits for this system?

Address bus size (m) = number of bits required to encode virtual memory

$$\text{Index virtual @} = 2 \text{ Byte}$$

$$2^m = \frac{\text{Size of virtual memory}}{\text{index V@}} = \frac{2^{33}}{2} = 2^{32} @ \Rightarrow m = 32 \text{ bits} \quad (0.5 \text{ pt})$$

Exercise 6 : (Replacement Algorithms) (4.5 Points / 15 Minutes)

Consider the following page reference string: 7, 5, 2, 3, 1, 1, 7, 2, 0, 3, 6, 5, 4, 3, 2

Assuming demand paging with four frames, how many page faults would occur for the following replacement algorithms?

a) FIFO

FIFO (1.5 pts)															
Reference String	7	5	2	3	1	1	7	2	0	3	6	5	4	3	2
Frame 1	<u>7</u>	7	7	7	<u>1</u>	1	1	1	1	1	1	<u>5</u>	5	5	5
Frame 2		<u>5</u>	5	5	5	5	<u>7</u>	7	7	7	7	7	<u>4</u>	4	4
Frame 3			<u>2</u>	2	2	2	2	2	<u>0</u>	0	0	0	0	<u>3</u>	3
Frame 4				<u>3</u>	3	3	3	3	3	3	<u>6</u>	6	6	6	<u>2</u>
Page Fault	X	X	X	X	X		X		X		X	X	X	X	X
Page Fault = 12															

b) LRU (Least Recently Used)

LRU (1.5 pts)															
Reference String	7	5	2	3	1	1	7	2	0	3	6	5	4	3	2
Frame 1	<u>7</u>	7	7	7	1	1	1	1	1	<u>3</u>	3	3	3	3	3
Frame 2		<u>5</u>	5	5	5	5	<u>7</u>	7	7	7	<u>6</u>	6	6	6	<u>2</u>
Frame 3			<u>2</u>	2	2	2	2	2	2	2	2	<u>5</u>	5	5	5
Frame 4				<u>3</u>	3	3	3	3	<u>0</u>	0	0	0	<u>4</u>	4	4
Page Fault	X	X	X	X	X		X		X	X	X	X	X		X
Page Fault = 12															

c) Second-Chance

Second-Chance (1.5 pts)															
Reference String	7	5	2	3	1	1	7	2	0	3	6	5	4	3	2
Frame 1	<u>7</u> ₁ ⁺	7 ₁ ⁺	7 ₁ ⁺	7 ₁ ⁺	<u>1</u> ₁ ⁺	1 ₁	1 ₁	1 ₁	1 ₁ ⁺	1 ₀	<u>6</u> ₁	6 ₁	6 ₁ ⁺	6 ₁ ⁺	<u>2</u> ₁
Frame 2		<u>5</u> ₁	5 ₁	5 ₁	5 ₀ ⁺	5 ₀ ⁺	<u>7</u> ₁	7 ₁	7 ₁	7 ₀	7 ₀ ⁺	<u>5</u> ₁	5 ₁	5 ₁	5 ₀ ⁺
Frame 3			<u>2</u> ₁	2 ₁	2 ₀	2 ₀	2 ₀ ⁺	2 ₁ ⁺	2 ₀	<u>3</u> ₁	3 ₁	3 ₁ ⁺	3 ₀	3 ₁	3 ₀
Frame 4				<u>3</u> ₁	3 ₀	3 ₀	3 ₀	3 ₀	<u>0</u> ₁	0 ₁ ⁺	0 ₀	0 ₀	<u>4</u> ₁	4 ₁	4 ₀
Page Fault	X	X	X	X	X		X		X	X	X	X	X		X
Page Fault = 12															

Good Luck