University of M'sila
Faculty of Mathematics and Computer Science
Department of Computer Science

جامعة المسيلة
كلية الرياضيات و الإعلام الآلي
قسم الإعلام الآلي

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

Duration: 1h 30 - Documents: not allowed

Academic year : 2025 / 2026

Level : L2 Semester : 5

Date : 10/05/2026 (08 : 30 – 10 : 30)

First Name : ……

Last Name : ……

Group : ……

Score : … / 20

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

Q1) 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)**

Q2) 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)**

Q3) 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)**

Q4) 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.

Q5) 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)**

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 **256 bytes**, what is the page number and offset in decimal?

A	F	F	9
1010	1111	1111	1001
$2^n = 256 \Rightarrow n = 8$			
1010 1111		1111 1001	
Page # = 175 (0.5 pts)		**Offset = 249** (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) < 1, 1 048 >

(0x0400) = 1 024
1 048 > 1 024, so, Illegal reference (error) (0.5 pts)

b) < 3, 1 048 >

0x1100 = 4 352 & 0x3200 = 12 800
1 048 < 4 352, so, Physical @ = 12 800 + 1 048 = 13 848 (0.5 pts)

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

a) 16 384

We have : 0x3200 = 12 800 and 0x1100 = 4 352
12 800 < 16 384 < 12 800 + 4 352 => 12 800 < 16 384 < 17 152
=> segment # = 3
Offset = 16 384 – 12 800 = 3 584
Logical @ = < 3, 3 584 > (0.5 pts)

b) 6 144

We have : 0x1400 = 5 120 and 0x1000 = 4 096
5 120 < 6 144 < 5 120 + 4 096 => 5 120 < 6 144 < 9 216
=> segment # = 0
Offset = 6 144 – 5 120 = 1024
Logical @ = < 0, 1024 > (0.5 pts)

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

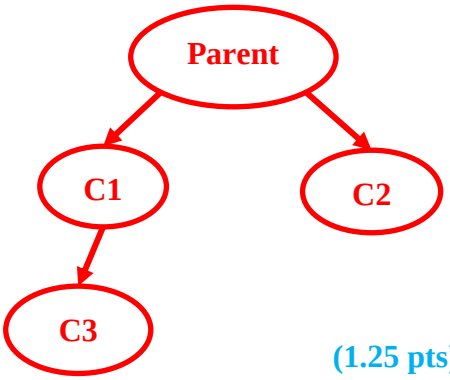
Consider the following section of the code and assume that all fork system calls are successful.

```
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.



(1.25 pts)

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	1st CPU	1st I/O	2nd CPU	2nd I/O	3rd CPU
0	**A**	2	3	2	3	2
2	**B**	4	1	4	-	-
3	**C**	2	1	2	-	-
6	**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**
Algorithm	**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 **8** ***Byte***

Answer the following questions, always justifying your answer:

Q1) What is the size of a page?

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

$$Page\ Size = 2^{16} \times 8 = 2^{19}\ Byte = 512\ KB$$ (0.25 pts)

Q2) How many frames does physical memory contain?

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

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

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

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

Q3) What is the size of physical memory?

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

We have : $frame\ size = page\ size = 512KB = 2^{19}\ Byte$

$$Size\ of\ Physical\ memory = 2^{15} \times 2^{19} = 2^{34} = 16\ GB$$ (0.5 pts)

Q4) What is the size of virtual memory?

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

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

$$Size\ of\ virtual\ memory = 2^{16} \times 2^{19} = 2^{35} = 32\ GB$$ (0.5 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

$$Index\ virtual\ @ = 8\ Byte$$

$$2^m = \frac{Size\ of\ virtual\ memory}{index\ V@} = \frac{2^{35}}{8} = 2^{32}@ \Rightarrow m = 32\ bits$$ (0.5 pt)

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

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

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	**2**	**5**	**7**	**3**	**1**	**1**	**2**	**7**	**0**	**3**	**6**	**5**	**4**	**3**	**7**
Frame 1	**2**	2	2	2	**1**	1	1	1	1	1	1	**5**	5	5	5
Frame 2		**5**	5	5	5	5	**2**	2	2	2	2	2	**4**	4	4
Frame 3			**7**	7	7	7	7	7	**0**	0	0	0	0	**3**	3
Frame 4				**3**	3	3	3	3	3	3	**6**	6	6	6	**7**
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	**2**	**5**	**7**	**3**	**1**	**1**	**2**	**7**	**0**	**3**	**6**	**5**	**4**	**3**	**7**
Frame 1	**2**	2	2	2	1	1	1	1	1	**3**	3	3	3	3	3
Frame 2		**5**	5	5	5	5	**2**	2	2	2	**6**	6	6	6	**7**
Frame 3			**7**	7	7	7	7	7	7	7	7	**5**	5	5	5
Frame 4				**3**	3	3	3	3	**0**	0	0	0	**4**	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	**2**	**5**	**7**	**3**	**1**	**1**	**2**	**7**	**0**	**3**	**6**	**5**	**4**	**3**	**7**
Frame 1	$\mathbf{2_1^+}$	2_1^+	2_1^+	2_1^+	$\mathbf{1_1}$	1_1	1_1	1_1	1_1^+	1_0	$\mathbf{6_1}$	6_1	6_1^+	6_1^+	$\mathbf{7_1}$
Frame 2		$\mathbf{5_1}$	5_1	5_1	5_0^+	5_0^+	$\mathbf{2_1}$	2_1	2_1	2_0	2_0^+	$\mathbf{5_1}$	5_1	5_1	5_0^+
Frame 3			$\mathbf{7_1}$	7_1	7_0	7_0	7_0^+	7_1^+	7_0	$\mathbf{3_1}$	3_1	3_1^+	3_0	3_1	3_0
Frame 4				$\mathbf{3_1}$	3_0	3_0	3_0	3_0	$\mathbf{0_1}$	0_1^+	0_0	0_0	$\mathbf{4_1}$	4_1	4_0
Page Fault	X	X	X	X	X		X		X	X	X	X	X		X
Page Fault = 12															

Good Luck