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 : 2024 / 2025

Level : L2 Semester : 5

Date : 18/05/2025 (12 : 30 – 14 : 00)

First Name : ……

Last Name : ……

Group : ……

Score : … / 20

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

Q1) What are the two main objectives of an operating system? Please briefly elaborate them.

1) **To provide user with the convenience**, that is to provide an environment for users to execute programs on computer hardware in a safe convenient, protected and efficient manner; **(0.5 pts)**

2) **It does resource allocation**: to allocate resources in a fair and efficient manner. **(0.5 pts)**

Q2) There are three types of schedulers. Which are they?

- Long-term scheduler, **(0.25 pts)**
- Short-term schedulers and **(0.25 pts)**
- Medium-term schedulers. **(0.25 pts)**

Q3) How does aging solve the starvation problem in priority based scheduling system?

It **works by increasing the priority of a process** as time passes and **will eventually be the highest priority request** (after it has waited long enough) **(0.5 + 0.5 = 1 pts)**

Q4) How do base register and limit register help in protecting against illegal memory access?

Any access to memory is **checked against these two registers**. **(0.25 pts)**

The address should be **greater or equal than the base address** and **smaller than (Base register + limit register)**. **(0.25 + 0.25 = 0.5 pts)**

Q5) How does external fragmentations occur? How could we resolve this problem?

External fragmentation exists when there is enough **total memory space to satisfy a memory request**, but **the available spaces are not contiguous**; storage is fragmented into a large number of small holes, each of which may be insufficient to satisfy the request. **(0.75 pts)**

Compaction technique can be used to solve this problem, but it is costly **(0.25 pts)**

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

Consider the page table for a system with **12 bits** virtual and physical addresses and **512 bytes** pages

Page #	Frame #	Presence/Absence bit
0	7	0
1	2	1
2	4	0
3	5	1
4	6	0
5	4	1
6	3	1
7	0	1

Q1) How many bits is the offset coded on?

$$Page\ size = 512 = 2^{Offset} \Rightarrow Offset = 9\ bits$$ **(0.75 pts)**

Q2) Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in **hexadecimal**.

a) 7EF

$Virtual\ @ = 7EF$	
0111 1110 1111	
$Page\ \#$	$Offset$
011	1 1110 1111
Page # = 3 $\xrightarrow{In\ Page\ Table}$ P/A bit = 1, so => Frame # = 5	
$Frame\ \#$	$Offset$
101	1 1110 1111
1011 1110 1111	
$Physical\ @ = BEF$	**(0.75 pts)**

b) 111

$Virtual\ @ = 111$	
0001 0001 0001	
$Page\ \#$	$Offset$
000	1 0001 0001
Page # = 0 $\xrightarrow{In\ Page\ Table}$ P/A bit = 0,	
$Physical\ @\ cannot\ be\ known$	**(0.75 pts)**

c) 2FF

$Virtual\ @ = 2FF$	
0010 1111 1111	
$Page\ \#$	$Offset$
001	0 1111 1111
Page # = 1 $\xrightarrow{In\ Page\ Table}$ P/A bit = 1, so => Frame # = 2	
$Frame\ \#$	$Offset$
010	0 1111 1111
0100 1110 1111	
$Physical\ @ = 4FF$	**(0.75 pts)**

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

A parent process (**PID = 100**) with the characteristics, described in the table below, creates a child process (**PID = 200**) by using the system call **`fork()`**.

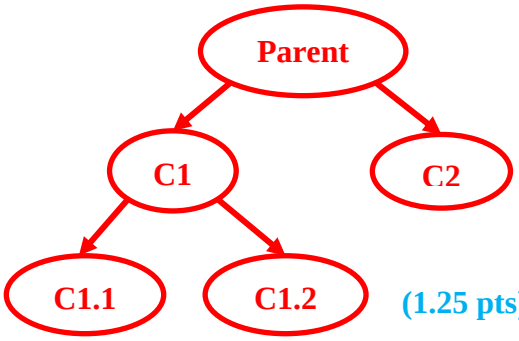
Q1) Enter the four missing values into the table.

	Parent Process	Child Process	
PPID	**99**	**100**	**(0.25 pts)**
PID	**100**	**200**	
UID	**25**	**25**	**(0.25 pts)**
Return code of `fork()`	**200**	**0**	**(0.25 + 0.25 = 0.5pts)**

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

```
#include <stdio.h>
#include <unistd.h>
int main()
{
    if (fork() || fork())
        fork();
    printf("1 ");
    return 0;
}
```

Q2) Draw a process tree for the above program.



(1.25 pts)

Q3) What will be printed?

1 1 1 1 1 **(0.25 pts)**

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**	1	3	1	3	1
2	**B**	3	2	3	-	-
3	**C**	3	2	3	-	-
6	**D**	2	1	2	1	2

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**) **(1.5 pts)**

CPU	A		B	B	B	C	C	C	A	D	D	B	B	B	C	C	C	A	D	D		D	D		
QUEUE				C	C	A	A	A	D	B	B	C	C	C	A	A	A	D							
					A		D	D	B		C			A	D	D	D								
								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		B	B		C	C	A	A	A	D							D				
QUEUE										A		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**) **(1.5 pts)**

CPU	A		B	B	C	C	A	B	D	D	C	A	B	B	D	D	B	C	C	D	D	C			
QUEUE				C	A	A	B	D	C	C	A			D	B	B	C	D	D	C	C				
					B	B	D	C								C									
							C																		

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				A	A	A	B	B	D	C	C		D								
QUEUE									B	B	D	D	C												
												C													

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. **(1.5 pts)**

	Turnaround Time (TR)				**Average**	**Waiting Time (WT)**				**Average**
Algorithm	**A**	**B**	**C**	**D**		**A**	**B**	**C**	**D**	
FCFS	**18**	**12**	**14**	**17**	**15,25**	**15**	**6**	**8**	**11**	**10**
RR with Quantum=2	**12**	**15**	**19**	**15**	**15,25**	**9**	**9**	**13**	**9**	**10**

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

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

- A size of page table is $\mathbf{128\ KBytes}$
- The offset is coded on $\mathbf{16\ bits}$
- Number of page table entries equal to $\mathbf{65536}$
- A page table entry is of the form: $\boldsymbol{n}$ | **1** | **3** where :
  - ✓ $\boldsymbol{n}$ is the number of bits to code a frame
  - ✓ **1** is the absence/presence bit
  - ✓ **3** is the number of bits to encode the page load date
- A virtual address is indexed by $\mathbf{1\ Byte}$

Answer the following questions, always justifying your answer:

Q1) Find the value of $\boldsymbol{n}$.

✓ $Size\ of\ PT = \#\ of\ PT\ entries \times PT\ entry\ size \Rightarrow PT\ entry\ size = \frac{Size\ of\ PT}{\#\ of\ PT\ entries}$

✓ $\Rightarrow PT\ entry\ size = \frac{128\ KBytes}{65536} = \frac{2^{17}}{2^{16}} = 2\ Bytes = 16\ bits$ **(0.25 pts)**

✓ $PT\ entry\ size = \#\ of\ bits\ to\ code\ a\ frame + (1\ bit)P/A\ bit + (3\ bits)\ PLD$

✓ $Size\ of\ a\ PT\ entry = n + 1 + 3 = n + 4 \Rightarrow n = Size\ of\ a\ PT\ entry - 4$

✓ $\Rightarrow n = 16 - 4 = 12\ bits$ **(0.25 pts)**

Q2) How many frames does physical memory contain?

✓ $\#\ of\ frames = 2^{\#\ of\ bits\ to\ code\ a\ frame} = 2^{n}$ **(0.25 pt)**

✓ $\#\ of\ frames = 2^{12}\ Frames$ **(0.25 pt)**

Q3) What is the size of a page?

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

$Page\ Size = 2^{16} \times 1 = 2^{16}\ Bytes = 64\ KB$ **(0.25 pts)**

Q4) What is the size of physical memory?

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

We have : $frame\ size = page\ size = 2^{16}\ Bytes = 64\ KB$

$Size\ of\ Physical\ memory = 2^{12} \times 2^{16} = 2^{28} Bytes = 256\ MB$ **(0.5 pts)**

Q5) What is the size of virtual memory?

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

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

$Size\ of\ virtual\ memory = 2^{16} \times 2^{16} = 2^{32} Bytes = 4\ GB$ **(0.5 pts)**

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

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

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	**1**	**2**	**3**	**4**	**5**	**6**	**3**	**2**	**3**	**7**	**3**	**6**	**5**	**2**	**1**
Frame 1	**1**	1	1	1	**5**	5	5	5	5	**7**	7	7	7	7	7
Frame 2		**2**	2	2	2	**6**	6	6	6	6	6	6	**5**	5	5
Frame 3			**3**	3	3	3	3	**2**	2	2	2	2	2	2	**1**
Frame 4				**4**	4	4	4	4	**3**	3	3	3	3	3	3
Page Fault	X	X	X	X	X	X		X	X	X			X		X
Page Fault = 11															

b) LRU (Least Recently Used)

LRU (1.5 pts)															
Reference String	**1**	**2**	**3**	**4**	**5**	**6**	**3**	**2**	**3**	**7**	**3**	**6**	**5**	**2**	**1**
Frame 1	**1**	1	1	1	**5**	5	5	5	5	**7**	7	7	7	**2**	2
Frame 2		**2**	2	2	2	**6**	6	6	6	6	6	6	6	6	6
Frame 3			**3**	3	3	3	3	3	3	3	3	3	3	3	**1**
Frame 4				**4**	4	4	4	**2**	2	2	2	2	**5**	5	5
Page Fault	X	X	X	X	X	X		X		X			X	X	X
Page Fault = 11															

c) Second-Chance

Second-Chance (1.5 pts)															
Reference String	**1**	**2**	**3**	**4**	**5**	**6**	**3**	**2**	**3**	**7**	**3**	**6**	**5**	**2**	**1**
Frame 1	$\mathbf{1_1^+}$	1_1^+	1_1^+	1_1^+	$\mathbf{5_1}$	5_1	5_1	5_1^+	5_1^+	$\mathbf{7_1}$	7_1	7_1	7_1^+	7_0	7_0
Frame 2		$\mathbf{2_1}$	2_1	2_1	2_0^+	$\mathbf{6_1}$	6_1	6_1	6_1	6_0^+	6_0^+	6_1^+	6_0	$\mathbf{2_1}$	2_1
Frame 3			$\mathbf{3_1}$	3_1	3_0	3_0^+	3_1^+	3_0	3_1	3_0	3_1	3_1	3_0	3_0^+	$\mathbf{1_1}$
Frame 4				$\mathbf{4_1}$	4_0	4_0	4_0	$\mathbf{2_1}$	2_1	2_0	2_0	2_0	$\mathbf{5_1}$	5_1	5_1^+
Page Fault	X	X	X	X	X	X		X		X			X	X	X
Page Fault = 11															

Good Luck