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)

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.

……
……
……
……

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

……
……
……

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

……
……
……
……

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

……
……
……

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

……
……
……
……
……

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?

……

……

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

a) 7EF

……

……

……

……

……

b) 111

……

……

……

……

……

……

c) 2FF

……

……

……

……

……

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**	…
PID	**100**	**200**
UID	**25**	…
Return code of `fork()`	…	…

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.

……

……

……

……

……

……

Q3) What will be printed?

……

……

……

……

……

……

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**)

CPU																									
QUEUE																									

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																									
QUEUE																									

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**)

CPU																									
QUEUE																									

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																									
QUEUE																									

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.

	Turnaround Time (TR)				**Average**	**Waiting Time (WT)**				**Average**
Algorithm	**A**	**B**	**C**	**D**		**A**	**B**	**C**	**D**	
FCFS	…	…	…	…	…	…	…	…	…	…
RR with Quantum=2	…	…	…	…	…	…	…	…	…	…

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

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

- A page table of size $\mathbf{128}\ \boldsymbol{KByte}$
- The offset is coded on $\mathbf{16}\ \boldsymbol{bits}$
- 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 $\mathbf{1}\ \boldsymbol{Byte}$

Answer the following questions, always justifying your answer:

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

……

……

……

……

Q2) How many frames does physical memory contain?

……

……

……

……

Q3) What is the size of a page?

……

……

……

Q4) What is the size of physical memory?

……

……

……

Q5) What is the size of virtual memory?

……

……

……

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														
Reference String	**1**	**2**	**3**	**4**	**5**	**6**	**3**	**2**	**3**	**7**	**3**	**6**	**5**	**2**	**1**
Frame 1															
Frame 2															
Frame 3															
Frame 4															
Page Fault															
Page Fault = ⋯															

b) LRU (Least Recently Used)

LRU														
Reference String	**1**	**2**	**3**	**4**	**5**	**6**	**3**	**2**	**3**	**7**	**3**	**6**	**5**	**2**	**1**
Frame 1															
Frame 2															
Frame 3															
Frame 4															
Page Fault															
Page Fault = ⋯															

c) Second-Chance

Second-Chance														
Reference String	**1**	**2**	**3**	**4**	**5**	**6**	**3**	**2**	**3**	**7**	**3**	**6**	**5**	**2**	**1**
Frame 1															
Frame 2															
Frame 3															
Frame 4															
Page Fault															
Page Fault = ⋯															

Good Luck