Lebanese University
Faculty of Sciences I

Second Session 2020/2021
Duration: 90 minutes

I3303 - INFO324
Operating System II

Problem I — **20 points**

Consider a virtual memory with page size of 1024 bytes, 8 virtual pages and 4 physical pages. Its page table is as follows:

virtual page	physical page
0	3
1	1
2	null
3	null
4	2
5	null
6	0
7	null

1. Give the List the addresses that cause a page fault. (Starting address and ending address)
2. Convert the following virtual address to physical address: 0, 3727, 1023, 1024, 7425, and 4196.

Problem II — **40 Points**

Part A: Consider a hard drive with a capacity of 256 GB and a file management system using FAT-32.

1. What is the minimum size of a physical block given that each entry in FAT table is 28 bits?
2. Calculate the number of blocks needed to store the FAT table on disk.

Part B: We propose to study a variant of UNIX file system where the inode contains 8 fields with double level of indexing. Each field contains the index of a map block of level1 which contains also 1024 index of map blocks (level 2). Each block of level 2 may contain 1024 index of data blocks.

1. What is the size of a block in this system? justify
2. How many bytes the number of a block occupies? justify
3. What is the maximum number of blocks of maps (first and second level)? Justify.
4. What is the maximum size (in bytes) of a file supported by this FS? Justify.

83886080

Problem III (20+15) 35 ~~30~~ points

A) What is the possible output of the following two programs? Justify your answer while taking into account all possible scenarios and identifying if a possible deadlock can occur.

20

Program 1

```
void main()
{
        int i, j = 1, p[2];
        pipe(p);
        write(p[1],&j,sizeof(int));
        for(i=1;i<5;i++)
                if(!fork()){
                        close(p[1]);
                        break;
                }
        read(p[0],&j,sizeof(int));
        printf("%d\n",i);
}
```

Program 2

```
void main()
{
        int i, j = 1, p[2];
        pipe(p);
        write(p[1],&j,sizeof(int));
        for(i=1;i<5;i++)
                if(!fork()){
                        close(p[1]);
                        break;
                }
        wait(0);
        read(p[0],&j,sizeof(int));
        printf("%d\n",i);
}
```

B) Draw the tree of processes generated by the following code:

15

```
int main(void)
{
        ( fork() && fork() ) || fork() && ( fork() && ( fork() || fork() ) );
        sleep(2);
        Return 0;
}
```

Pb I (25 pts) 30

page size = 1024 bytes

1) - page fault on page 2 ⇒ [2048 - 3071]
   - page fault on page 3 ⇒ [3072 - 4095]
   - page 5 ⇒ [5120 - 6143]
   - page 7 ⇒ [7168 - 8191]

(10)

2) Convert to physical :

virtual	Physical
* 0	3 × 1024 = 3072 (4)
3727	1679

3727 = 3 × 1024 + 655 ⇒ on virtual page 3 (2)

- virtual page 3 is not loaded ⇒ page fault
- eject page from memory : suppose physical frame 1 ⇒ 3727 = 1 × 1024 + 655 = 1679 (6)

* 1023 ⇒ on virtual page 0 ⇒ physical frame 3

⇒ 1023 = 3 × 1024 + 1023 = 4095 (6) (4)

* 1024 = 1 × 1024 + 0 ⇒ 1024 = 1 × 1024 + 0 = 1024 (W) (6)

* 7425 = (7 × 1024 + 257) ⇒ 7425 page fault (2)

* 4196 = (4 × 1024 + 100) ⇒ 4196 = 2 × 1024 + 100 (1)

(W) = 2148 (6)

Pb II : (35 pts) (15+20)

HD : 256 GB

FAT32

1) Size of physical block?

each entry is 28 bits

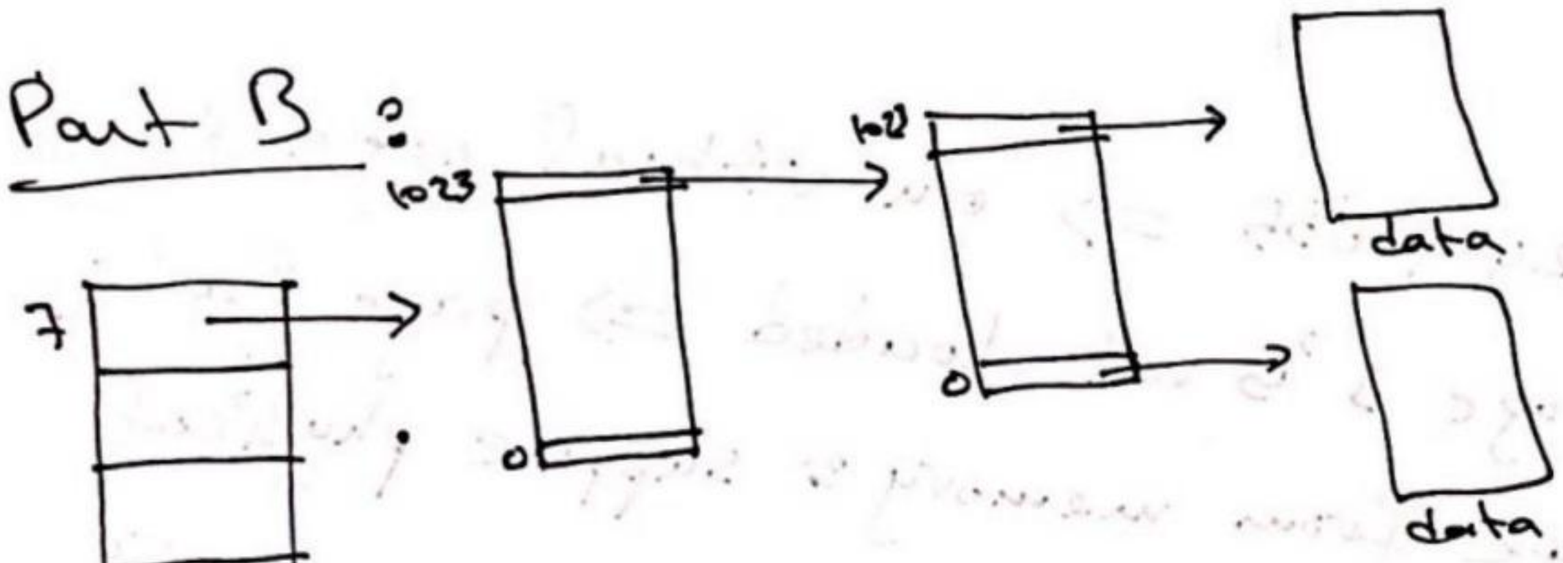
we have 2^{28} blocks ⇒ size of block is (2)

$$\frac{256\,GB}{2^{28}} = \frac{2^8 \times 2^{30}}{2^{28}} = \frac{2^{38}}{2^{28}} = 2^{10} \text{ bytes}$$

2) Size of FAT is $2^{28} \times 28$ (2)

nb of blocks is : $\frac{2^{28} \times 28}{2^{10}} = 28 \times 2^{18}$ blocks

Part B :



1) Size of block?

(5) the total nb of data blocks is : $8 \times 1024 \times 1024$

$= 2^3 \times 2^{10} \times 2^{10} = 2^{23}$ blocks

$\Rightarrow$ size(block) $= \frac{256\,GB}{2^{23}} = \frac{2^{38}}{2^{23}} = \boxed{2^{15} \text{ bytes}}$

(5) 2) nb of byte $= \frac{2^{15}}{1024} = \frac{2^{15}}{2^{10}} = 2^5 = \boxed{32}$

(5) 3) max nb of map blocks? $8 \times 1024 = \underline{2^{13} \text{ blocks}}$

(5) 4) max size of file ?? $= (8 \times 1024 \times 1024) - 8 \times 1024$

$= (8 \times 1024)(1024 - 1) = 8 \times 1024 \times 1023$

3 (35 = 20 + 15)

A) Prog1

* if the parent reads first and exit, then all child will print also : 5 4 3 2 1
(because the parent exits and the write side on child is closed)

* If any child read first, it prints the value (ex 4). All other processes including the part will block because they will attempt to read from an empty pipe with write side opened.

B) Prog2

* if any of child read first from pipe. (suppose child 1) ⇒ then it will print 1 and exits
* any other child will be blocked because he attempts to read from empty pipe with write side opened from the parent until the parent exit by executing wait(0)

* ~~when the parent execute wait, it prints 5~~
* ~~after all other child will print (i)~~

the parent will be blocked because he didn't close its write side (2) and all other child will be blocked also

(10)

(10)

1	1	1	1	1	1	1
5	5	5	5	5	5	5
2	3	4	4	4	3	2
3	2	2	3	2	4	4
4	4	3	2	3	2	3

all other combinations are possible for other child ex : child 2 can starts first etc...

B)