LEBANESE UNIVERSITY
Faculty of Sciences – Section I
Info 3303-324 - E

Final 2018-2019
Duration: 2 hours

Part I — Process Management — 20 Points

1. Draw the tree of processes and show all possible outputs by the execution of the following C program under UNIX

```
void main(){
fork();
if(fork() + fork())
        printf("%d\n", fork() || fork());
}
```

2. Write a C program where the parent process creates N child processes P1, P2, …, PN (in this order). After creation, the child processes start writing without stopping and alternatively their PIDs in a common such that at a given instant the content of the pipe is similar to the following figure (PID(i) is the id of Pi)

PID(x) PID(x-1) … PID(2)PID(1) … PID(N) … PID(2) PID(1) PID(N) … PID(2) PID(1) PID(N) … PID(2) PID(1)

Meanwhile, the parent process must be in waiting state for a time t. when he awakes, he must kill its child and display the pid of the last child was written its pid in the pipe (PID(x) in the figure above)

Part II — File Management — 25 Points

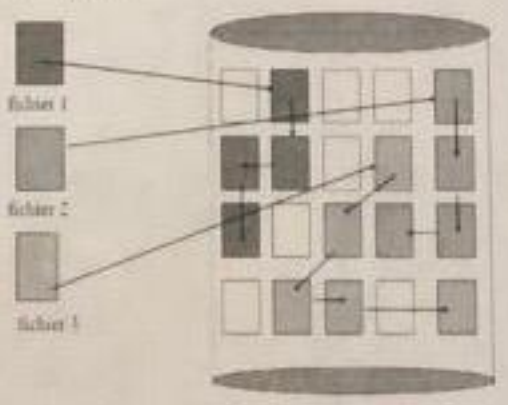
The following parts are independents:

A) Given a FS where the topo contains 15 entries:
   a. The first 12 entries refer directly to data blocks.
   b. The entry 12 points to a map block (1 level of indirection)
   c. The entry 13 corresponds to two levels of indirection.
   d. The entry 14 corresponds to three levels of indirection

The size of the block is 1 KB and the number of a block occupies 4 bytes:
   a. What is the maximum number of map block in the system? Justify your answer.
   b. We consider a file with 300,000 bytes. How many blocks (data and maps) are necessary for representing this file on disk? Justify your answer.

B) A process reads sequentially a file with size 37MB at a rate of 256 bytes at a time. Assume that the size of the block is 512 bytes and a block number occupy 4 bytes. Moreover, the average disk access time is 12ms.
   1) The system does not provide disk caching mechanism, i.e., that each read request requires disk I/O operation Give the total number of disk accesses required and the total I/O disk access time.
   2) Now, the system maintains a cache mechanism in the disk, which stores in main memory 1000 disk blocks most recently accessed. Give the total number of disk accesses required and the total I/O disk access time

C) The following diagram represents the FAT allocation of disk C on a Windows system. The blocks are numbered from 1 to 20, line to line, from left to right.



Represent the contents of the FAT table according to the allocation shown in the diagram. The blocks are numbered from 1 per line from left to right.

```c
#include <stdio.h>
#include <unistd.h>
#include<stdlib.h>
#include<sys/wait.h>
#include<signal.h>
#include<sys/types.h>

int p[2],pid;
void handler();


void main(){
int i, mypid, n=5;
pipe(p);

while(1){
signal(SIGALRM, handler);
for (i=0;i<n;i++)
{
  if(!fork()){
     mypid=getpid();
     while(1){
      write(p[1],&mypid,sizeof(int));
      sleep(n+i);
     }
  }
}

alarm(2);
pause();
}
}

void handler(){
signal(SIGALRM, handler);
printf("in handler\n");
while(read(p[0],&pid,sizeof(int)))
{
 printf("the pid to kill is:%d\n",pid);
 kill(pid,SIGKILL);
}
printf("the last pid written in the pipe is: %d\n",pid);
}
```

Parent : 6 (fork + handling)
child : 2 (writing into pipe)
general : 2 (variable)

6 + 2 + 2

27

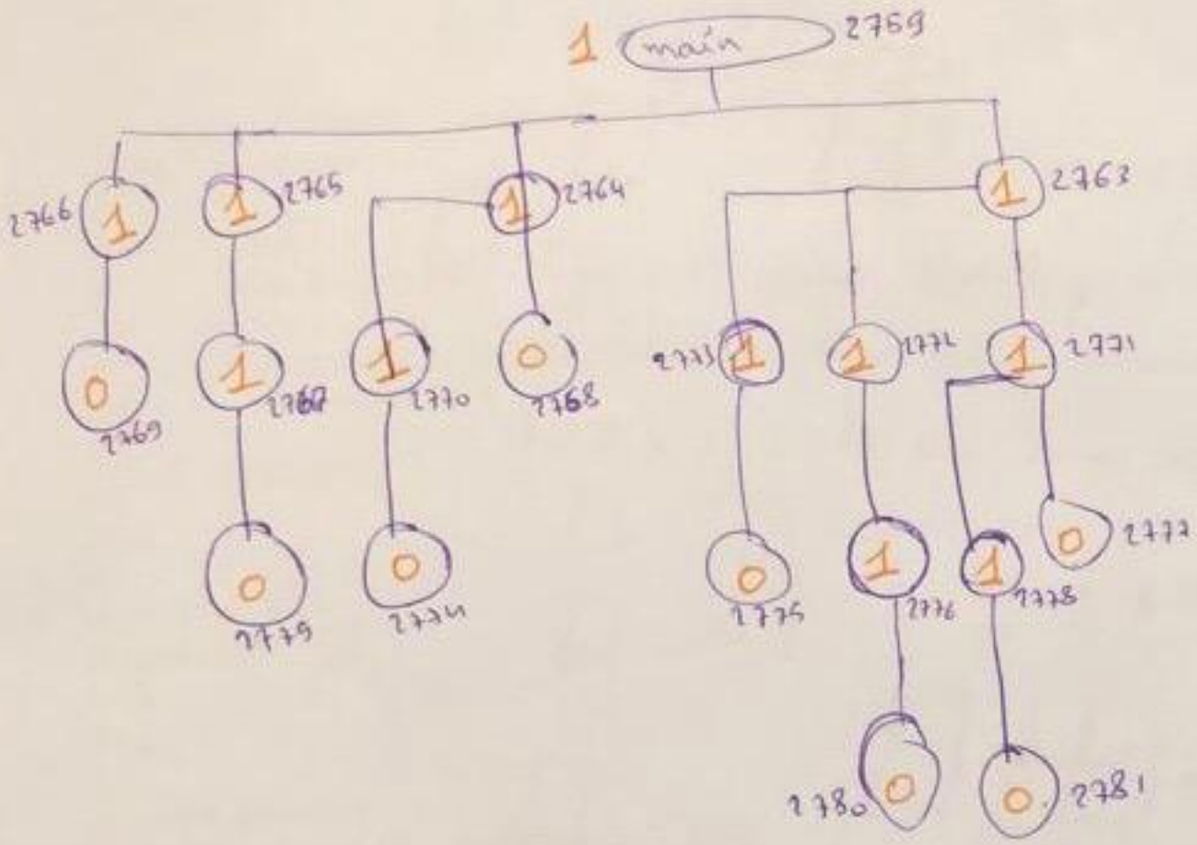
I3303 - Solution

2018-2019 (Final)

Part I (20 pts)

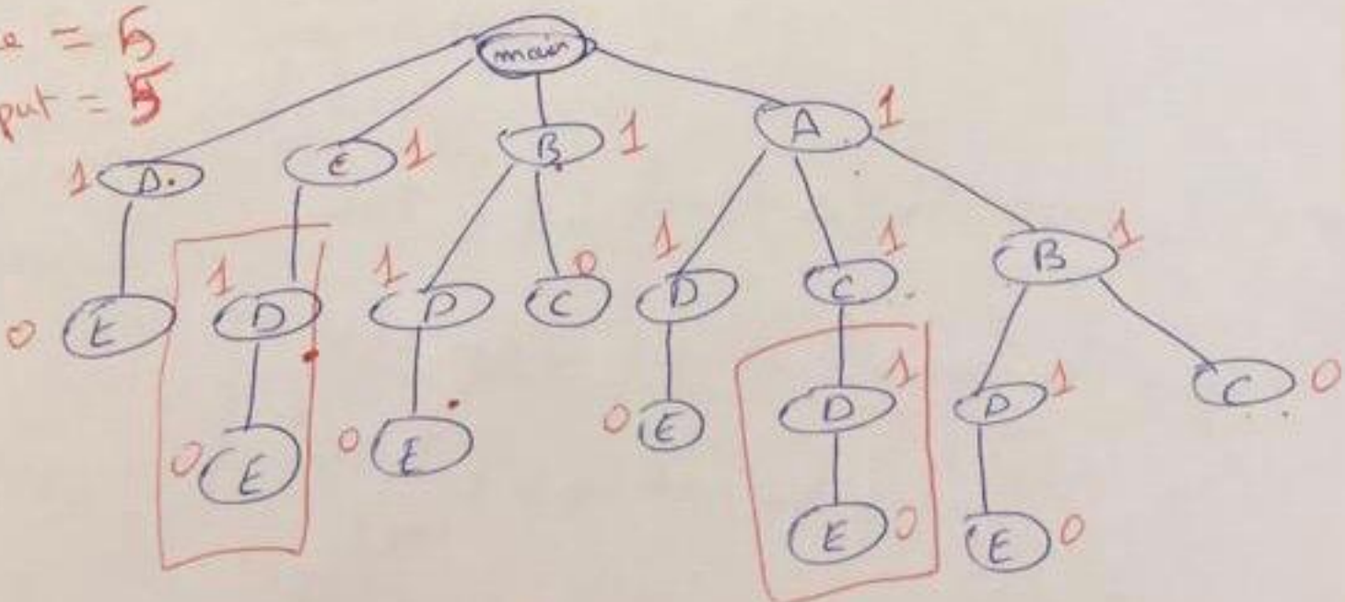
1) (10 pts)

```
void main()
{
A // fork();
  if (fork() + fork())
B       C
      printf("%d\n", fork() || fork());
                     D      E
}
```



20 processes

* Tree = 5
* output = 5



Part II (continue)

B) 2) with caching mechanism

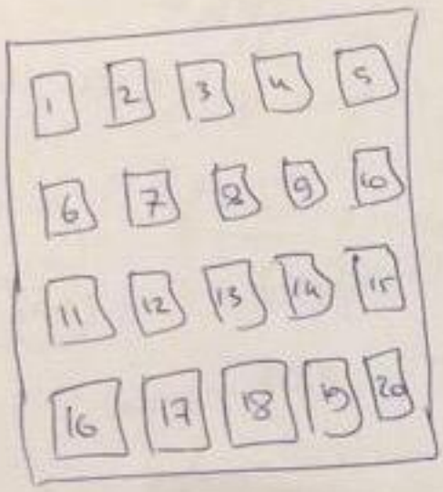
- each block saves 2 I/O requests
- each block is ~~fetched~~ loaded one time in memory

=> nb I/O access = nb I/O blocks

= 2 × 37 × 1024 I/O access (4)

=> time = 2 × 37 × 1024 × 12 ms

C)

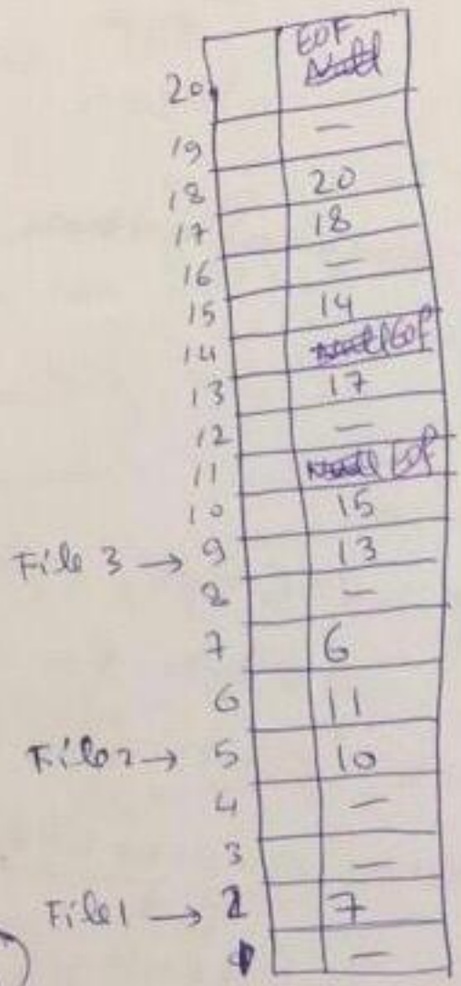


(8)

File 1 : 2 → 7 → 6 → 11

File 2 : 5 → 10 → 15 → 14

File 3 : 9 → 13 → 17 → 18 → 20



(3)

Part III

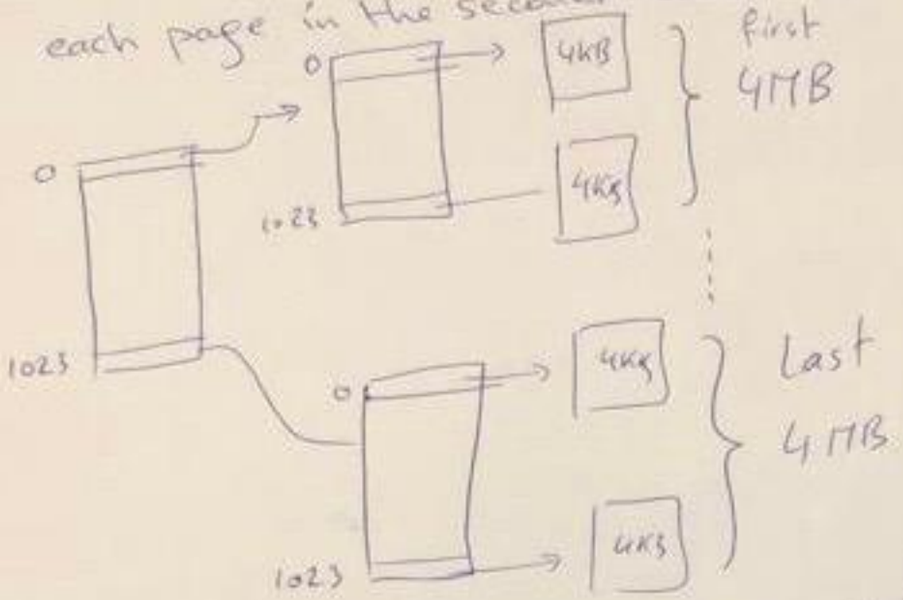
ii) process size = 22 MB (code, data, stack) (3)

code → [2MB - 6MB]

data → [12MB - 21MB]

Pages that contains PTE's of these two segments

* each page in the second level covers 4MB of virtual address space



* for the code segment we need 2 pages of level 2 (the first two)

* for the data segment, we need 3 pages of level 2 (pages: 4, 5, 6)

iii) * pure pagination (3)

* nb of entries ??

size physical memory = 1TB = 2^{20}

frame size = 4KB = 2^{12}

$\Rightarrow$ nb of frames = $\frac{2^{20}}{2^{12}} = 2^8$

$\Rightarrow$ we need 2^8 entries in the inverted page table

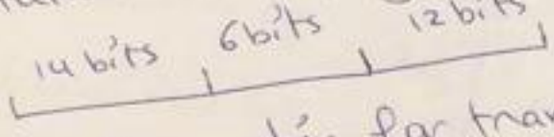
Part III (continued)

c) So the physical address is
(2604 + 20) + 209 out of address
~~segmentation fault~~

(4) because the physical memory is at max 2048 frames

B) virtual addressing in 32 bits (14 pts)
Page size = 4 KB
physical memory = 1 MB

a) segmentation with pagination



missing information for translation: AE854C9C ??

solution: the missing information is the segment descriptor table (SDT) (2)

b) steps to translation

- Given the SDT, from the first 14 bits we get the nb of segment in SDT (segment entry)
- from the segment entry, we get the (corresponding) page table of this segment
- the second 6 bits are the page # (PTE) in the page table
- knowing the PTE, we get the frame #
- frame # + offset (12 bits) => physical address

(3)