THEORY OF INFORMATION, ARCHITECTURE OF COMPUTERS AND OPERATING SYSTEMS (TIACOS)
Bioinformatics, ESCI
DAC, UPC

Final exam **Spring 2022**

- This exam is **open book**, **closed Internet**. You can use your laptop, but do not connect to the Internet until you are told to do so.
- Write your answers in the companion files:
  1. `ShortQuestions.txt`
  2. `Processes.py`
  3. `Counting.py`
  4. `FileSystem.txt`
- Time **94 minutes**. You will have 8 minutes at the end of the test to upload your tar file to `aula.esci.upf.edu`
- Grades will be published at aula.esci on **Sunday, 3 July** evening. Review for the exam will be held online via Google Meet on **Monday, 4 July**, 10.00 - 11.00 am.

* * *

1. (3 points) **Short questions**
   (a) The kernel takes control of the system against three events: interrupts, system calls and ... What is the third?

   exceptions

   (b) How many open file tables are in memory? And inode tables?

   File tables, only one. Inode tables, only one.

   (c) What's the name of the fixed-size block in which logical memory is divided? If we have 10 bits to address blocks, how many blocks of memory we can address?

   Page. You can address $2^{10} = 1024$ blocks.

   (d) Considerer a preemptive Round Robin scheduler, with a 10 cycles quantum. Answer the questions with True or False:
      1. When a process gets assigned the CPU, it allways gets a whole execution quantum.

         True

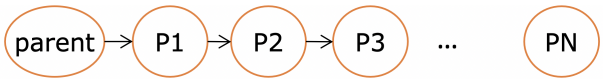
      2. A process can leave the CPU after consuming 7 cycles.

         True

   (e) What do the COW acronyms mean? What system call starts it?

Copy On Write. It's started by the `fork()` system call.

2. (3 points) **Processes.py**. Write a Python program that creates `N` processes, where $0 \leq \texttt{N} \leq 255$. You can see the result in the following image (a circle is a "process" and the arrow means "process creation"). Note that there is never more than one process in run state and each process creates at most one process. In the end, it prints the list of all the created pids ($[P_1, \ldots, P_N]$). Let's assume there are not exceptions during the execution of this code.




```
l=[]
for _ in range(N):
  pid=os.fork()
  if pid > 0:
    os.wait()
    sys.exit(0)
  else:
    l.append(os.getpid())
print(l)
```

3. (2 points) **Counting.py**. Read the following Python code carefully and answer the questions. Let's assume there are less than 255 cytosines and there are not exceptions during the execution of this code. This code should count the number of cytosines in the fastq file `SRR000049.seqs`. For some reason, it does it twice. Make **as few changes as possible** so that the result is correct, and it's printed once. You must keep the concurrency.

```
1  nCs = 0
2  pid = os.fork()
3  fd = os.open("SRR000049.seqs",os.O_RDONLY)
4  c = os.read(fd,1)
5  while (len(c)>0):
6     if (c == b'C'):
7        nCs +=1
8     c = os.read(fd,1)
9  print ("Number of C:",nCs)
10 os.close(fd)
```

Assuming there are less than 255 cytosines:

```
1     nCs = 0
2     fd = os.open("SRR000049.fastq",os.O_RDONLY)
3     pid = os.fork()
4     c = os.read(fd,1)
5     while (len(c)>0):
6        if (c == b'C'):
7           nCs +=1
8        c = os.read(fd,1)
9     if (pid>0):
10     status = os.wait()
11    else:
12     sys.exit(nCs)
13    print ("Number of C:",nCs+(status[1]>>8))
14    os.close(fd)
```

4. (2 points) **File system**. In a Linux system, this is screen output from terminal. The current working directory is `/homeA/j/jfornes/examen/`

```
jfornes@sert-entry-2:~/examen$ ls -liaR
.:
total 28
22289205 drwxr-xr-x 3 jfornes ac 4096 Nov 28 09:20 .
22282340 drwx--x--x 82 jfornes ac 12288 Nov 28 10:27 ..
22289207 -rw-r--r-- 3 jfornes ac 5 Nov 24 13:52 idem.txt
22289207 -rw-r--r-- 3 jfornes ac 5 Nov 24 13:52 one_file.txt
22289214 lrwxrwxrwx 1 jfornes ac 1 Nov 24 12:27 quiz -> .
22289249 drwxr-xr-x 2 jfornes ac 4096 Nov 28 10:27 subdir
22289215 prw-r--r-- 1 jfornes ac 0 Nov 24 12:29 talk_to_me.ch
./subdir:
total 24
22289249 drwxr-xr-x 2 jfornes ac 4096 Nov 28 10:27 .
22289205 drwxr-xr-x 3 jfornes ac 4096 Nov 28 09:20 ..
22289283 -rwxr-xr-x 1 jfornes ac 8568 Nov 28 10:27 e.out
22289251 lrwxrwxrwx 1 jfornes ac 32 Nov 28 10:14 quatre_file.txt ->
/homeA/j/jfornes/examen/idem.txt
22289207 -rw-r--r-- 3 jfornes ac 5 Nov 24 13:52 tres_file.txt
```

(a) (0.5 points) Complete the inode table answering theese two questions:

1. Which is the type of the inode 22289215?

   Inode 22289215 is a pipe.

2. Which is the number of links of the inode 22289207?

Inode 22289207 has 3 links.

Inode	2289205	22282340	22289207	22289214	22289249	22289215	22289251	22289283
# links	3	82	**?**	1	2	1	1	1
Type	d	d	-	l	d	**?**	l	-

(b) (0.5 points) List all the file names of the hard links to access the inode 22289207.

Inside the parent directory `/homeA/j/jfornes/examen/` it has three file names:

`one_file.txt`
`idem.txt`
`tres_file.txt`

Also, the symlink `subdir/quatre_file.txt`

(c) (1 point) Given the following source code that corresponds to the Python script `fs.py` and, assuming that we launch the program from the shell, with no system call returns an error, **reasonably answer the questions**.

```
1 fd = os.open("/homeA/j/jfornes/examen/subdir/quatre_file.txt",os.O_RDONLY)
2 s = os.lseek(fd,0,os.SEEK_END)
3 print(s)
4 os.close(fd)
```

1. What inodes does the execution of line 1 read? Write down the inode number when it is known.

   It has to read every inode of the directories in the path, that is, "/" (the inode number 2), "homeA", "j"(we cannot guess the inode number from the statement of this last two), "jfornes"(22282340"), "examen" (22289205) and "subdir" (22289249). After that, it reads inode of quatre_file.txt (22289251) that is a softlink and it forces to read the inode of "idem.txt" (22289207).

2. What data blocks does the execution of line 1 read?

   It has to read every data block of the directories in the path, that is, "/", "homeA", "j", "jfornes", "examen" and "subdir". But it has not need to read the data block of "quatre_file.txt" because the pathname size of "idem.txt" fits in the inode 22289251, already read.

3. What is the value of the variable "fd" after the execution of line 1?

   As the program is launched from the shell, "fd" will be 3. In other words, the file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

4. What is the value of variable "s" after the execution of line 2?

   5. Upon successful completion, lseek() returns the resulting offset location as measured in bytes from the beginning of the file. In this case, the file size that is also returned by the "ls - liaR" command line showed at the beginning of this statement.