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

Final exam **Spring 2021**

Grades will be published at aula.esci on **Thursday, 1 July** evening.

Review for the exam will be held in building C6, room 216, Campus Nord, UPC on **Friday, 2 July**, 9.00 - 10.00 am.

1. (2 points) **Short questions**
   (a) The kernel takes control of the system against three events: interrupts, exceptions and ... What is the third?

   system calls

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

   File descriptors tables, one per process. File tables, only one.

   (c) What's the name of the fixed-size block in which physical memory is divided? If we have 32 blocks of memory, how many bits you need to address them?

   Frame. You need 5 bits to address 32 blocks.

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

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

2. (4 points) **Processes**. Read the following Python code carefully and answer the questions. Let's assume there are not exceptions during the execution of this code[1].
   (a) (2 points) At most, how many processes are running concurrently? How many are parent processes? How many are child processes?

   4 processes, 1 parent process and 3 children processes at most.

   (b) (0.5 points) Which kind of process, parent or child, will execute the function `work()`?

   Child.

   (c) (0.5 points) Which kind of process, parent or child, will execute the function `handler()`?

[1] This code works in Python 2.7 and Python 3.8.

```
1  count = 3
2  def handler(signum, frame):
3     global count
4     [pid, status] = os.waitpid(-1, os.WNOHANG)
5     count = count - 1
6     if count == 0 :
7        sys.exit(0)
8     return 0
9  def work():
10    os.execlp("uname","uname")
11    return 2
12 signal.signal(signal.SIGCHLD, handler)
13 i = 0
14 while i < count:
15    pid = os.fork()
16    if pid == 0:
17       work()
18    i = i + 1
19 while True:
20    pass
```

Parent.

(d) (1 point) Executing line 4, could block the process? And line 12?

Line 4 no, because it uses `WNOHANG`. Line 12, no it just program a signal.

3. (4 points) **Files**. Read the following Python code carefully and answer the questions. Let's assume there are not exceptions during the execution of this code[2].

```
1  fd = os.pipe()
2  out = os.open("OUT.txt",os.O_CREAT|os.O_TRUNC|os.O_RDWR,0o644)
3  pid = os.fork()
4  if ( pid == 0):
5     os.close(fd[0])
6     buff="EAE"
7     i=0
8     while (i<3):
9        os.write(fd[1],buff[i])
10       i = i + 1
11    os.close(fd[1])
12 else:
13    os.close(fd[1])
14    buff="XMN"
15    i = 0
16    c = os.read(fd[0],1)
17    while (i<3):
18       os.write(out,c)
19       os.write(out,buff[i])
20       c = os.read(fd[0],1)
21       i = i + 1
22    os.close(fd[0])
23    os.wait()
```

[2]This code only works in Python 2.7.

(a) (0.5 points) How many pipes are created?

Just one pipe.

(b) (0.25 points) In the parent process PCB, how many file descriptors are added by this code?

3 entries in the file descriptor table, 2 because the pipe and 1 for the open.

(c) (0.25 points) In the child process PCB, how many file descriptors are added by this code?

It inherits as many as its parent had, because the fork system call.

(d) (0.5 points) How many bytes reads the parent process from the pipe?

3 bytes

(e) (0.5 points) After executing line 2, what are the permissions of the file "OUT.txt"

(f) (1 point) What's the final content of the file "OUT.txt"?

EXAMEN

(g) (0.5 points) If line 22 (`os.close(fd[0])` is removed, the final result will be the same or will any process block?

The result will be the same.

(h) (0.5 points) If line 13 (`os.close(fd[1])` is removed, the final result will be the same or will any process block?

The result will be the same.