THEORY OF INFORMATION, ARCHITECTURE OF COMPUTERS AND OPERATING SYSTEMS (TIACOS)
Bioinformatics, ESCI
Computer Architecture Dept., UPC
2020/2021 – 3th term

Unit 5 Input/Output. Exercises **Thu 3 Jun 2021**

1. The first parameter of `os.write(1,b'hello')`...

   △ represents a virtual device ×
   ◇ represents a logical device ✓
   ○ represents a physical device ×
   □ is wrong ×

2. The device driver ...

   ○ can execute privilege instructions ×
   △ implements access/control operations on physical devices ×
   ◇ implements a different interface for each OS ×
   □ all of the above ✓

3. The first parameter of `os.open("nom",os.O_WRONLY)`

   ○ represents a phsyical device ×
   △ represents a virtual device ×
   ◇ represents a logical device ×
   □ is wrong ✓

4. After executing the sequence: `ret=os.close(0);fd=os.open(fn,os.O_WRONLY)`

   △ `ret==-1` and `fd == -1` ×
   ○ `ret==0` and we cannot know the value of `fd` ×
   ◇ `ret==0` and `fd==0` ✓
   □ `ret==0` and `fd==-1` ×

5. Mark the wrong answer.

   A process is using the file `/etc/passwd`. Executing a new `open("/etc/passwd",O_WRONLY)`

   △ If there is no error, creates a new entry in the File Descriptor Table ×
   ○ If there is no error, creates a new entry in the File Table ×
   ◇ If there is no error, creates a new entry in the Inode table ✓
   □ In the File Table we will have at least 2 entries related to `/etc/passwd` ×

6. Mark the wrong answer.

   `fd` is a valid virtual device. Then `os.close(fd)`

   △ Always modifies the File Descriptor Table ×

   ○ Always modifies the Inode Table ✓

   ◇ Always modifies the File Table ×

   □ The inode in disk may be modified ×

7. The contents of the file named `nom` are "123". After executing `fd=os.open("nom",os.O_RDONLY); buf=os.read(fd,10)`

   △ `len(buf)==-1` and `buf` empty ×

   ○ `len(buf)==3` and `buf=="123"` ✓

   ◇ `len(buf)==-1` and `buf==123` ×

   □ `len(buf)==0` and `buf=="123"` ×

8. The file "nom" contains "123" and `buf==b'4'`. After executing `fd=os.open("nom",os.O_WRONLY); os.write(fd,buff)`

   △ `len(buf)==1` and `nom=="423"` ✓

   ○ `len(buf)==4` and `nom=="1234"` ×

   ◇ `len(buf)==-1` and `nom=="123"` ×

   □ `len(buf)==0` and `buf=="123"` ×

9. The file `nom` contains "hola". After executing `fd=os.open("nom",os.O_RDWR); os.read(fd,1); write(fd,b'2')`, `nom` contains ...

   △ "2ola" ×

   ○ hola2 ×

   ◇ h2la ✓

   □ 2hola ×

10. The following system call never blocks a process

    △ `pipe` ✓

    ○ `read` ×

    ◇ `open` ×

    □ `write` ×

11. Would the following system call block the process? `os.open("nom",os.O_RDWR)`

    △ Never ✓

    ○ Always ×

    ◇ Only if `nom` is a named pipe ×

    □ Only if `nom` is a file ×

12. Would the following system call block the process? `os.open("nom",os.O_WRONLY)`

    △ Never ×
    ○ Always ×
    ◇ Only if `nom` is a named pipe and there aren't readers on it ✓
    □ For all type of files, will block if there aren't readers ×

13. Could a process `A` blocked on a read receive a signal `SIGPIPE`?

    △ Yes, only if `A` is reading from a pipe without writers ×
    ○ No, because reading do not cause `SIGPIPE`s ×
    ◇ Yes, only if other process sends this signal to `A` ✓
    □ None of the above ×

14. A process execute the following code: `fd=os.pipe();c=os.read(fd[0],1)`

    △ The process will block forever in the read ✓
    ○ `read` will end returning `-1` because there are no writers ×
    ◇ `read` will end returning 0 because there are no writers ×
    □ None of the above ×

15. We create `A`, a soft link to `B`. Mark the **wrong** answer

    △ File `A` will have a new inode ×
    ○ if we delete `B`, the contents are deleted from disk ×
    ◇ result of `cat A` is the same as `cat B` ×
    □ This operation modifies the inode of `B` ✓

16. We create `A`, a hard link to `B`. Mark the **wrong** answer

    △ File `A` will have a new inode ✓
    ○ If we delete `B`, the contents are not deleted from disk ×
    ◇ The result of `cat A` is the same as `cat B` ×
    □ This operation modifies the inode of `B` ×

17. How many blocks accesses an `open` system call?

    △ Depends on the path of the file ✓
    ○ This system call does not accesses blocks ×
    ◇ Only the first block of the file ×
    □ None of the above ×

18. How many blocks accesses a `read` system call?

    △ Depends on the path of the file ×
    ○ This system call does not accesses blocks ×
    ◇ Only the inode of the file ✓
    □ None of the above ×