Assignment 3 (tentative due 24.06)

Understanding file management.

The goal of this lab is to build a better understanding about how files are treated in UNIX-like OS (we will look at the specific example in Linux). In particular, as you already should know each process maintains a table of open files (file descriptors), later these tables inside processes point to a global table of open files, where at least the current position in the file and the i-node number are maintained. The third table contains a table of inodes, the internal structure of inodes we discussed during the lecture.

To deal with the questions below, create a separate directory, where you copy all given files 1-5.c (the links can be found below near each question).

1.1 Create a directory named INODE. Inside create a file named **hello.txt**:

```
kirillk@OS20:~/OS20/HW3$ cat >> hello.txt
```

```
Hello world!
```

!!! Do not forget to press ctrl-D to create the end of the file.

To check the inode number (in red) run the following:

```
kirillk@OS20:~/OS20/HW3$ ls -li hello.txt
```

```
1310969 -rw-r--r-- 1 kirillk kirillk 6 Jun  9 15:35 hello.txt
```

I-node number

When a process opens a file, OS returns a file descriptor that the process uses later to access/change the file.

Copy into the previously created INODE directory (where you have created hello.txt) 1.c and compile it into a.out

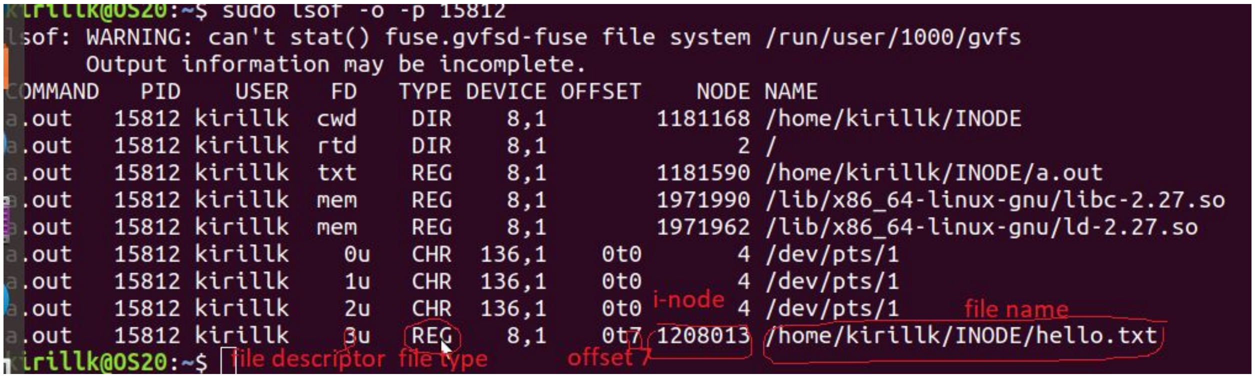
A. Review 1.c. In general this program opens the previously created file hello.txt containing and prints its process id, file name, and the file descriptor.

B. Open two terminals and change directory in both to INODE.

C. In one terminal run ./a.out and copy the printed pid.

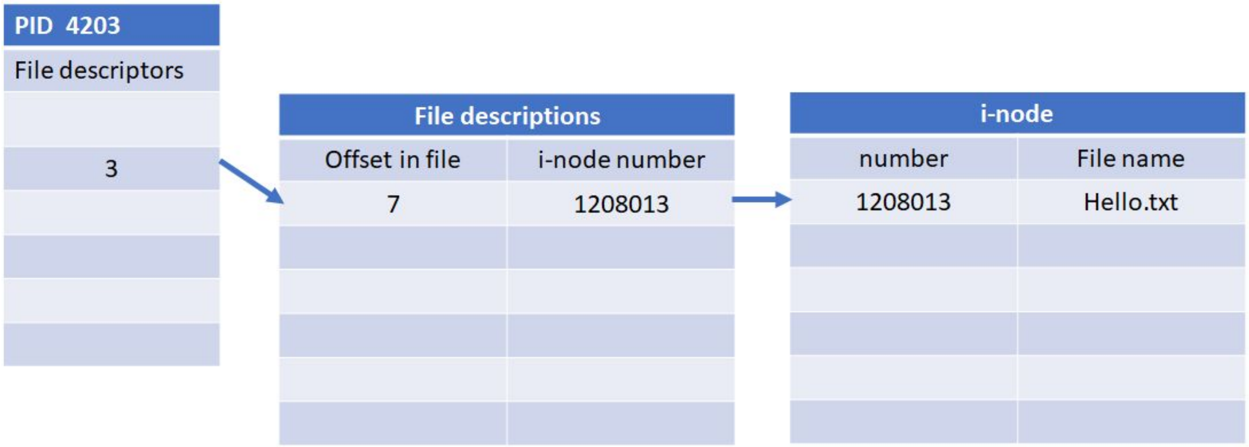
   ```
   kirillk@OS20:~/OS20/HW3/INODE$ ./a.out
   
   pid = 15812
   
   hello.txt, fd = 3
   ```

   In the second terminal run (do not forget sudo): `sudo lsof -o -p 15812`



```
kirillk@OS20:~$ sudo lsof -o -p 15812
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1000/gvfs
      Output information may be incomplete.
COMMAND    PID    USER   FD   TYPE DEVICE OFFSET    NODE NAME
a.out    15812 kirillk  cwd    DIR    8,1        1181168 /home/kirillk/INODE
a.out    15812 kirillk  rtd    DIR    8,1              2 /
a.out    15812 kirillk  txt    REG    8,1        1181590 /home/kirillk/INODE/a.out
a.out    15812 kirillk  mem    REG    8,1        1971990 /lib/x86_64-linux-gnu/libc-2.27.so
a.out    15812 kirillk  mem    REG    8,1        1971962 /lib/x86_64-linux-gnu/ld-2.27.so
a.out    15812 kirillk    0u   CHR  136,1    0t0       4 /dev/pts/1
a.out    15812 kirillk    1u   CHR  136,1    0t0       4 /dev/pts/1
a.out    15812 kirillk    2u   CHR  136,1    0t0       4 /dev/pts/1
a.out    15812 kirillk    3u   REG    8,1    0t7 1208013 /home/kirillk/INODE/hello.txt
kirillk@OS20:~$
```

As you can see: 3 means file descriptor 3, **u -** read/write (**r-** if read or **w** if write) t7 means that the current offset in hello.txt is 7 (check lseek in 1.c), also you can see the i-node number 1208013.



PID 4203
File descriptors
3

File descriptions	
Offset in file	i-node number
7	1208013

i-node	
number	File name
1208013	Hello.txt

Table 1

As you can see the table of open files contains a file descriptor 3 for hello.txt, this entry points to the global table of open files for all processes containing at least i-node number 1208013 and the current position 7 in the file. Later each such entry points to the third table with i-nodes containing at least i-node number and the file name. For simplicity we use only the local name and not the full name.

In the upcoming questions you will encounter various scenarios. Given the output of the lsof command add its output to the specific places and feel the tables. You will understand when separate entries are created in the first and the second tables (from the left).

Question 1.1 (25 points): What happens if we duplicate a file descriptor by dup().

dup() duplicates a file descriptor (see man dup for more details).

Fill Table 2 after running 2.c Review its code before.

PID <your pid>
File descriptors

File descriptions	
Offset in file	i-node number

i-node	
number	File name
	hello.txt

Table 2

Put here screenshot of lsof output

Question 1.2 (25 points): Fill Table 3 below for the case when the same process opens the same file twice as in 3.c. Review its code. Add the output of lsof to the dedicated place and given the values from lsof feel the table.

PID <your pid>
File descriptors

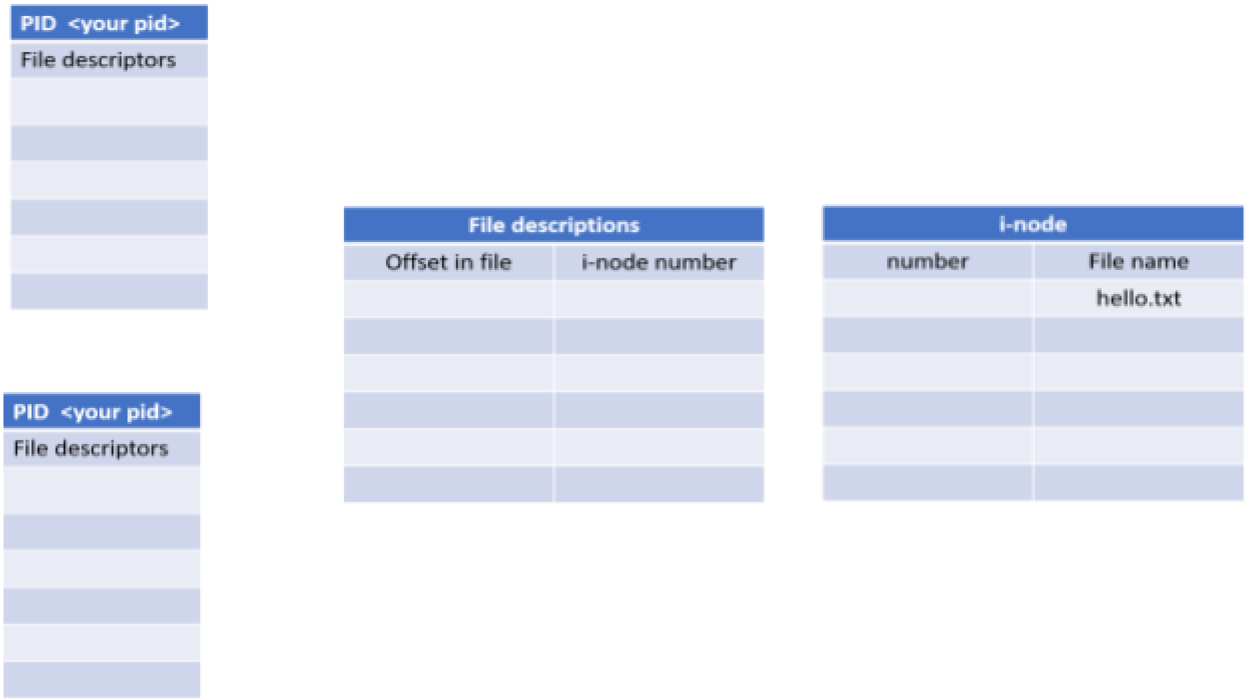
Table

File descriptions	
Offset in file	i-node number

i-node	
number	File name
	hello.txt

Put here screenshot of lsof output

Question 1.3 (25 points): Fill Table 4 below if a process opens a file and later fork() as in 4.c. Review the code and grab the output of lsof from both processes and based on the values feel the table



PID <your pid>
File descriptors

PID <your pid>
File descriptors

File descriptions	
Offset in file	i-node number

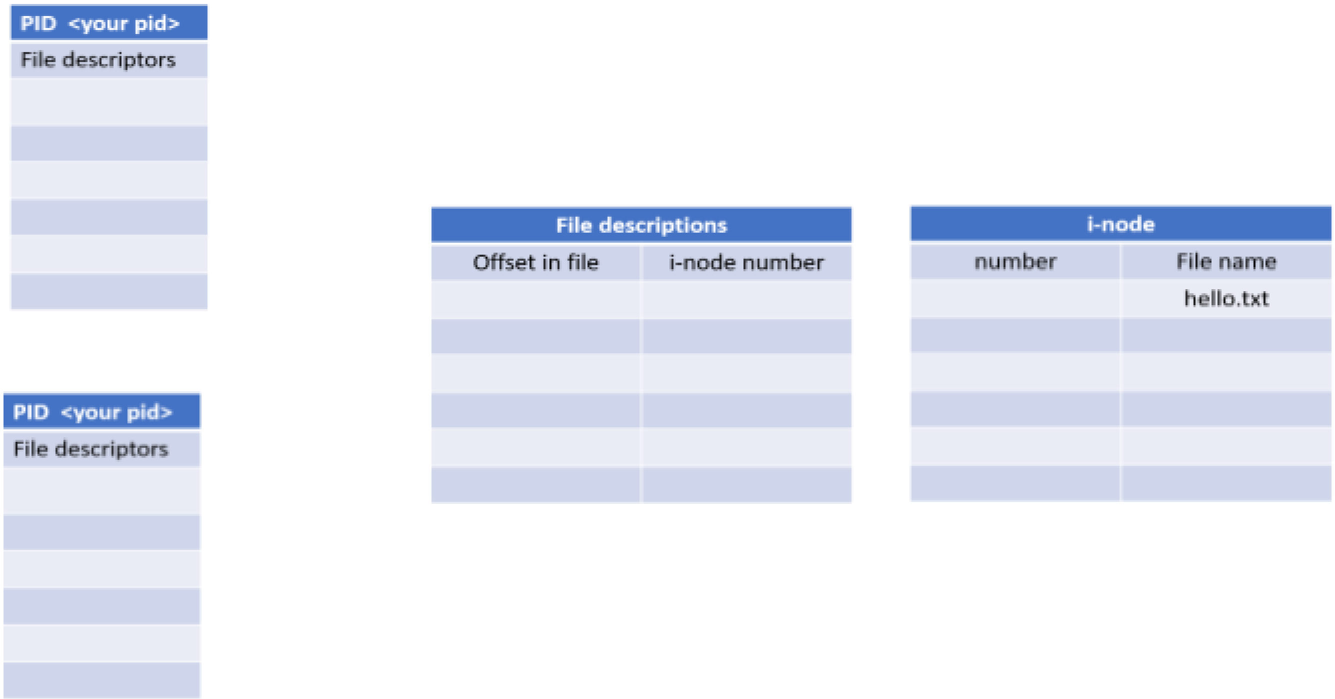
i-node	
number	File name
	hello.txt

Table 4

Put here screenshot of lsof output for the parent process.

Put here screenshot of lsof output for the child process.

Question 1.4 (25 points): Fill Table 5 when two processes as in 5.c open the same file: one run with ./a.out 4 , the other with ./a.out 6. 4 and 6 are the values for lseek in each process. Exit both processes only after you grab lsof from both processes, so use 3 terminals for this purpose.



PID <your pid>
File descriptors

PID <your pid>
File descriptors

File descriptions	
Offset in file	i-node number

i-node	
number	File name
	hello.txt

Table 5

Put here a screenshot of lsof output for the first process with lseek 4

Put here a screenshot of lsof output for the second process with lseek 6