15-213: Introduction to Computer Systems
Written Assignment #10

This written homework covers System Level I/O and Network Programming

Directions

Complete the question(s) on the following pages with single paragraph answers. These questions are not meant to be particularly long! Once you are done, submit this assignment on Canvas.

Below is an example question and answer.

Q: Please describe benefits of two's-complement signed integers versus other approaches.

A: Other representations of signed integers (ones-complement and sign-and-magnitude) have two representations of zero (+0 and −0), which makes testing for a zero result more difficult. Also, addition and subtraction of two’s complement signed numbers are done exactly the same as addition and subtraction of unsigned numbers (with wraparound on overflow), which means a CPU can use the same hardware and machine instructions for both.

Grading

Each assignment will be graded in two parts:

1. Does this work indicate any effort? (e.g. it’s not copied from a homework for another class or from the book)
2. Three peers will provide short, constructive feedback.

Due Date

This assignment is due on November 29th by 11:59 Pittsburgh time (currently UTC−4). Remember to convert this time to the timezone you currently reside in.

Question #1

Describe the changes made to the file descriptor table and open file table on calls to the process management syscalls of **fork**. What would happen if the refcount of an entry in the open file table becomes 0?

File descriptor table: When fork is called, the child process gets a copy of the file descriptor table from the parent process. The entries in the file descriptor table are shared between the parent and child, meaning they both initially point to the same file objects.

Open file table: When fork is called, the child process inherits a reference to the entries in the open file table from the parent process. The entries in the open file table are not duplicated; instead, the reference count for each entry is incremented.

If the refcount of an entry in the open file table becomes 0, it indicates that no process is using this file anymore. The operating system can then release the resources associated with that file, such as closing the file and freeing up memory.

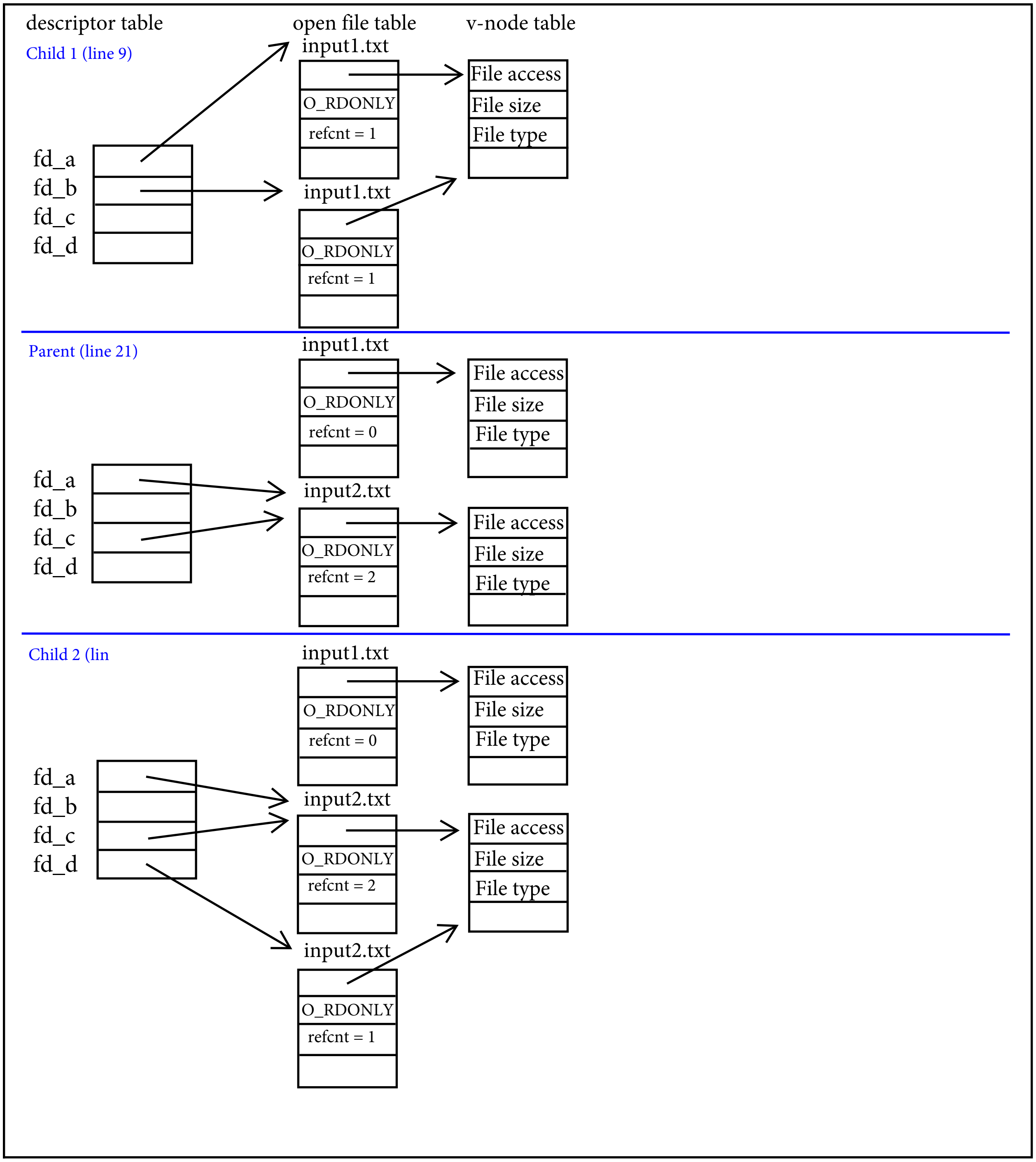
Question #2

Draw the descriptor table, open file table and v-node table when parent executes to line 21, child 1 executes to line 9, and child 2 executes to line 18.

You may assume that:

- fd_a = 2, fd_b = 3, fd_c = 3, fd_d = 4
- All system calls succeed

```
1       int main(int argc, char** argv)
2       {
3         int fd_a, fd_b, fd_c, fd_d;
4
5         fd_a = open("input1.txt", O_RDONLY);
6         if (fork() == 0) /* Child 1 */
7         {
8            fd_b = open("input1.txt", O_RDONLY);
9            ...
10        }
11        else
12        {
13           fd_c = open("input2.txt", O_RDONLY);
14           dup2(fd_c, fd_a);
15           if (fork() == 0) /* Child 2 */
16           {
17              fd_d = open("input2.txt", O_RDONLY);
18              ...
19           }
20        }
21        ...
22      }
```


descriptor table
open file table
v-node table
Child 1 (line 9)
fd_a
fd_b
fd_c
fd_d
input1.txt
O_RDONLY
refcnt = 1
input1.txt
O_RDONLY
refcnt = 1
File access
File size
File type
Parent (line 21)
fd_a
fd_b
fd_c
fd_d
input1.txt
O_RDONLY
refcnt = 0
input2.txt
O_RDONLY
refcnt = 2
File access
File size
File type
File access
File size
File type
Child 2 (lin
fd_a
fd_b
fd_c
fd_d
input1.txt
O_RDONLY
refcnt = 0
input2.txt
O_RDONLY
refcnt = 2
input2.txt
O_RDONLY
refcnt = 1
File access
File size
File type
File access
File size
File type

Question #3

Linux provides some powerful functions such as **getaddrinfo** and **getnameinfo** to convert between socket address data structure and string representation. Explain why **getaddrinfo** is generic enough to be used with any protocol?

getaddrinfo can be used with both IPv4 and IPv6 addresses, as well as other transport layer protocols. It abstracts away the details of different network protocols, allowing applications to work with various protocols without having to change their code. It can handle the translation between human-readable hostnames and IP addresses for both IPv4 and IPv6, making it versatile in modern networking environments where both address types are in use. getaddrinfo can resolve service names to port numbers, making it convenient for applications to specify services using human-readable names rather than having to hardcode port numbers. The function returns a linked list of addrinfo structures, each containing information about a particular address that matches the input criteria. This allows applications to iterate over the list and choose the appropriate address to use. getaddrinfo can also adapt to changes and developments of new protocols and web addresses.