Program 1A Report

Snapshots

Command execution:

```
[nithis13@csslab11 Program1a]$ ps -A | grep ssh | wc -l
3
[nithis13@csslab11 Program1a]$ g++ processes.cpp
[nithis13@csslab11 Program1a]$ ./a.out ssh
3
commands completed
[nithis13@csslab11 Program1a]$
```

```
[nithis13@csslab11 Program1a]$ ps -A | grep argv[1] | wc -l
0
[nithis13@csslab11 Program1a]$ g++ processes.cpp
[nithis13@csslab11 Program1a]$ ./a.out argv[1]
0
commands completed
[nithis13@csslab11 Program1a]$
```

Correctness:

```
[nithis13@csslab11 Program1a]$ ps -A | grep ssh | wc -l
3
[nithis13@csslab11 Program1a]$ g++ processes.cpp
[nithis13@csslab11 Program1a]$ ./a.out ssh
Grandchild process executing grep ssh
Great-grandchild process executing ps -A
Child process executing wc -l
3
Parent process
commands completed
[nithis13@csslab11 Program1a]$
```

Report on illustration of all processes and their pipe connection involved in processes.cpp

1) **How are processes (parent, child, grandchild, great grandchild)?**

In my code processes are created through fork system call and pipes. There are 4 processes, that's being created parent, child, grandchild and great grandchild. The first process that is created when program is started is parent processes, the parent process then creates a child process using fork, which takes care of the running wc –l. The child

process then creates a grandchild process which takes care of grep. The grandchild process then creates a great–grandchild which takes care of ps –A. All of the processes communicate with each other using pipes which connect their I/O.

2) **The name of each process**

Parent process executing and managing the overall pipeline, child executes wc –l command, grandchild executes grep command with the argument passed in ssh and lastly great-grandchild executes ps –A command.

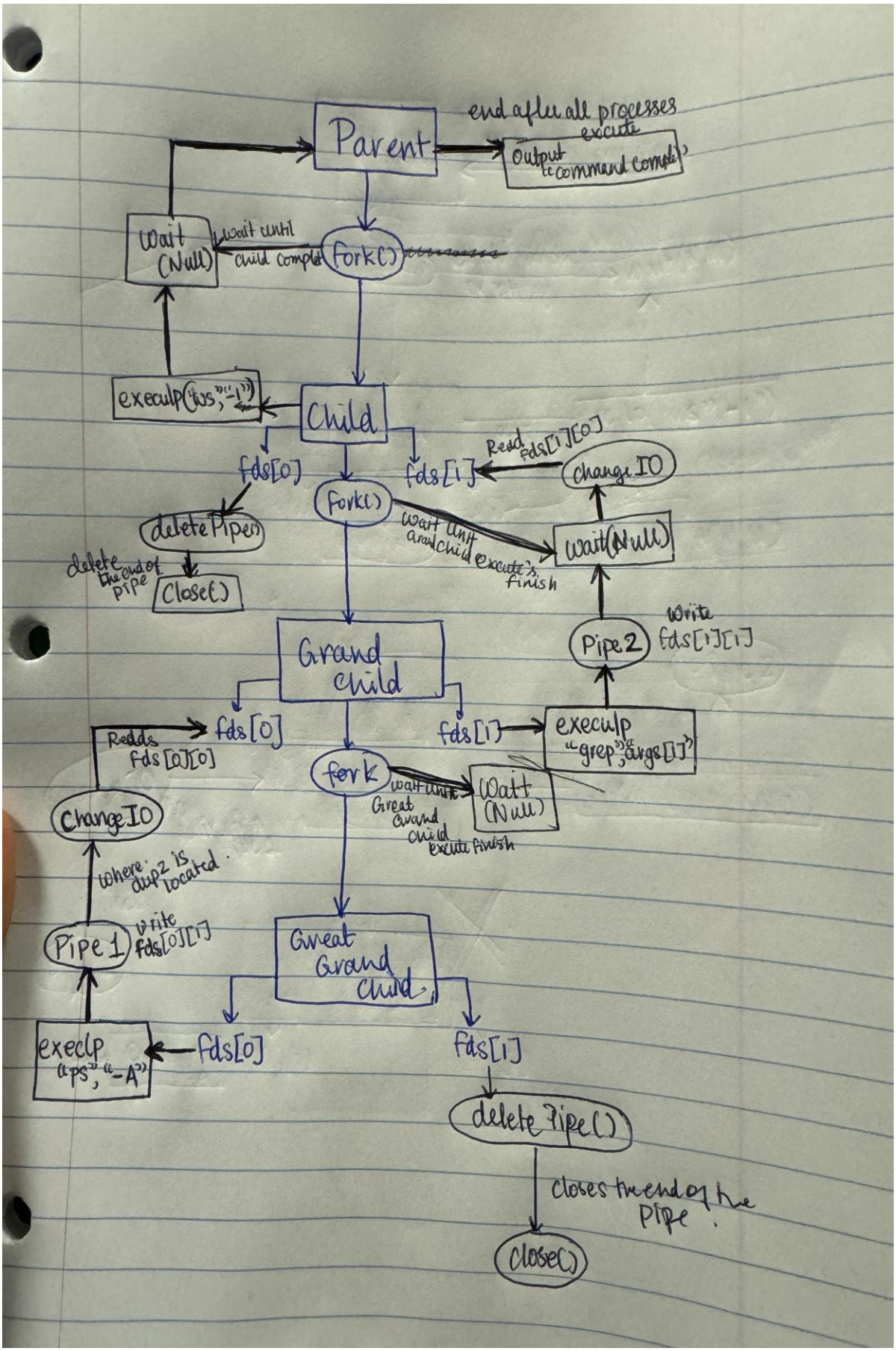
3) **How is each pipe connected?**

In this code, there are two pipes fds 0 and 1 which are created to connect process as indicated by const int SIZE = 2. Great-grandchild process writes output of ps –A to write end of first pipe fds[0][1] whereas grandchild process reads from read end of the first pipe fds[0][0] which contains the output from ps –A and it writes the output to write end of the second pipe fds[1][1] and lastly child process read from the read end of the second pipe fds[1][0] which contains output from grep.

4) **What file descriptor are opened or closed?**

Finally, the parent waits for the child process to finish and doesn't interact with pipes directly. Great-grandchild Process closes the read end of the first pipe and writes to the write end, closing both ends for the second pipe. Grandchild Process reads from the first pipe and writes to the second pipe, closing unused ends of both pipes. Child Process reads from the second pipe and writes to the terminal output.

DIAGRAM IN NEXT PAGE


Parent
end after all processes execute
Output "command complete"
Wait (Null)
wait until child complete
fork()
execulp("ls","-l")
Child
fds[0]
fds[1]
Read fds[1][0]
change IO
delete Pipe()
delete the end of pipe
close()
fork()
wait until grand child execute's finish
wait(Null)
Grand child
Write fds[1][1]
Pipe 2
Reads fds[0][0]
fds[0]
fds[1]
execulp "grep","args[1]"
fork
wait until Great Grand child execute finish
Wait (Null)
Change IO
where dup2 is located.
Pipe 1
Write fds[0][1]
Great Grand child
execlp "ps","-A"
fds[0]
fds[1]
delete Pipe()
closes the end of the pipe.
close()