Kevin Jiang

EID367

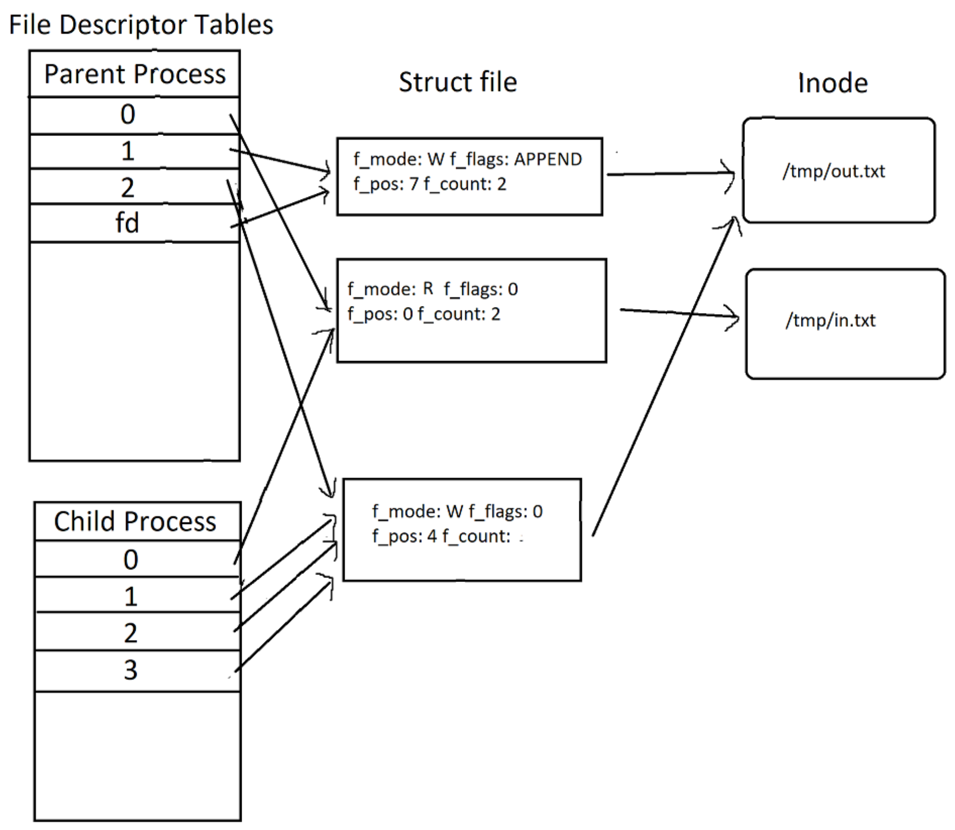
October 6th, 2020

Problem Set 3, Problems 1 and 2

1. Problem 1
   a. Data region, static modifier makes *i* a global variable, and it is initialized
   b. The function f1() is first called in the second line after main. For the parent process, f1() is called a second time after the if statement because the condition is false in the parent process. In the child process which is created when fork() is called, the return value of fork() is 0 and the f1() within the if statement will run. In the child process, the value of the integer *i* will start at 11, since fork() is called after the first f1() call in the parent process. The child process will call f1() a second time after the if statement. In total there are four f1() calls, so four lines of output.
   c. No, if there was an error in in the fork call, there will be only two lines of output. Errors are still possible, just very rare. Additionally, the order in which the parent and child processes run is indeterminate, so it is still possible for the child process to run first, resulting in a different output.
   d. The return value is 255. This makes sense because int ws = -1, which is 16 bits of 1. The least significant 8 bits may be changed to the exit status of the child process given by the wait() system call. But if you right shift ws by 8, the 8 least significant bits are replaced by 8 bits of 1. The 8 most significant bits will all be 0, so ws just becomes 255. 255 & 255 is just 255 again. That value is returned at the end of main().

2. Problem 2

   Computer generated output is “1234ABC” in the file out.txt. The <in.txt argument redirects standard input to in.txt, so for both the parent and the child processes, the original file descriptor that referred to stdin, which is 1, now refers to the struct file that points to in.txt. Since the struct file for stdin has no more references, it will be destroyed. >out.txt redirects standard output to out.txt, so the same thing happens with the struct file for stdout. 2>&1 redirects standard error to standard output, but since 1 was originally stdout and now refers to out.txt, 2 now also refers to out.txt. In the child process, 3 is a new file descriptor created using dup2 to refer to what 2 refers to, which is out.txt. In the parent process, fd is created to open out.txt in read-only append mode, so a new struct file also pointing to out.txt is created. Using dup2, 1 now also refers to the new struct file.


File Descriptor Tables
Parent Process
0
1
2
fd
Child Process
0
1
2
3
Struct file
f_mode: W f_flags: APPEND
f_pos: 7 f_count: 2
f_mode: R f_flags: 0
f_pos: 0 f_count: 2
f_mode: W f_flags: 0
f_pos: 4 f_count:
Inode
/tmp/out.txt
/tmp/in.txt