

Homework 5

Group Number 56 | Anmol Singhal 2017332, Daksh Shah 2017336, Tejas Oberoi 2017367

```
#include <stdio.h>
#include <unistd.h>
//ls | wc -l

/* File Descriptors Mapping
0 - STDIN
1 - STDOUT
2 - STDERR */

int main(){
    int pid; //Process ID

    int fd[2]; //(File Descriptors) To store the ends of the pipe, in
future
    /* fd[0] - Read End
       fd[1] - Write End*/

    int ret; //Return value

    /* Creates a pipe, and sets the values of the ends in fd variable
       returns -1 : when pipe cannot be created
           0 : when pipe is created successfully*/
    ret = pipe (fd);
    if (ret == -1) {
        printf ("Unable to create pipe\n");
        return 0;
    }

    pid = fork ();
    /* Tries to fork, i.e. to create a child process
       Two parallel executions (parent, child) start from the next line
       */

    if (pid == 0) { //Child; executes ls; writes to the pipe
        /* Verify that ls exists at /bin/ls */
        /* to verify run: which ls */
        char* const args[] = {"/bin/ls", NULL};
```

```

        close(fd[0]); //[^1] Closing the read end of the pipe

        close(1); //Closing STDOUT
        dup(fd[1]); //Duplicating Write End of pipe (to 1 - which is the
lowest FD available)

        close(fd[1]); //[^1] Closing write end of the pipe

        ret = execv (args[0], args);

        /* NOT REACHED*/
        printf ("failed to exec ls\n");
    } else if (pid > 0) { //Parent; executes wc; reads from the pipe
        /* Verify that wc exists at /usr/bin/wc */
        /* to verify run: which wc */
        char* const args[] = {"/usr/bin/wc", "-l", NULL};

        close(fd[1]); //Closing write end of the pipe
        /* Essential, Otherwise no output will come and it will be hung
in wait for more input
        Needed because, while reading from the pipe (done
in wc),
        it waits till it can get some input;
        i.e. Any threads have the write end of pipe open

        PS: after receiving EOF, it considers that
particular thread's write end to be closed
        */

        close(0); //Closing STDIN
        dup(fd[0]); //Duplicating Read end of pipe (to 0 - which is the
lowest FD available)

        close(fd[0]); //[^1] Closing read end of the pipe

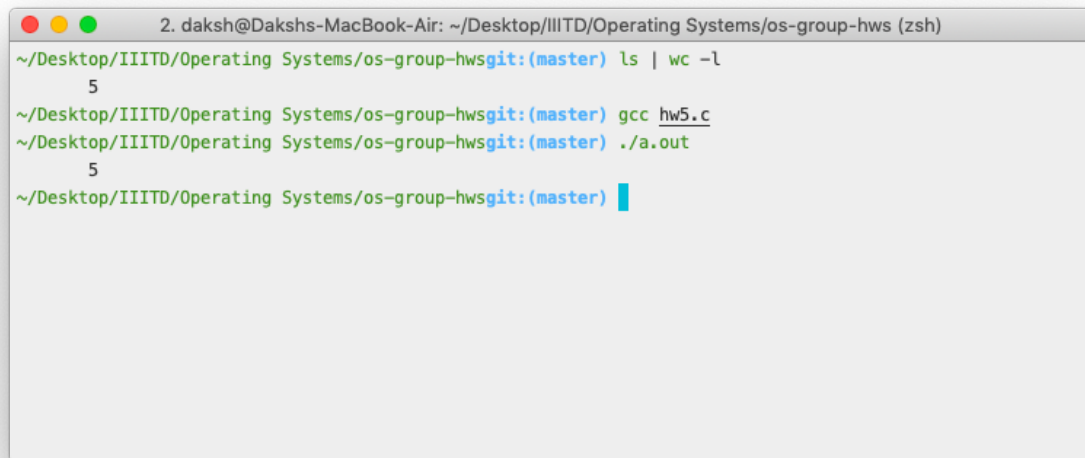
        execv (args[0], args);

        /* NOT REACHED*/
        printf ("failed to exec wc\n");
    } else { //Probably the fork() did not work properly
        printf ("Unable to fork\n");

```

```
}  
return 0;  
  
/*  
    [^1] : non-essential for the right output; done as a good coding  
practice  
    so that fd table entry can be used for other purposes  
*/  
}
```

Output:



```
2. daksh@Dakshs-MacBook-Air: ~/Desktop/IIITD/Operating Systems/os-group-hws (zsh)  
~/Desktop/IIITD/Operating Systems/os-group-hwsgit:(master) ls | wc -l  
5  
~/Desktop/IIITD/Operating Systems/os-group-hwsgit:(master) gcc hw5.c  
~/Desktop/IIITD/Operating Systems/os-group-hwsgit:(master) ./a.out  
5  
~/Desktop/IIITD/Operating Systems/os-group-hwsgit:(master) █
```