CSE344 – System Programming - Homework #1 Report
File I/O and file based interprocess communication

Student Name: Ozan Şelte

Student Number: 161044061

In this homework, I have not used any external temporary files for communication or any other purposes. Sometimes I used shorter system calls/functions which are not in our slides but course book to escape longer and hard to read functions. (For example: pread, pwrite instead of longer lseek lines; dup instead of fcntl). Finally, to be POSIX compatible I decided to define "_POSIX_C_SOURCE" as "200809" which is the latest POSIX standard.

Fast Fourier Transform code is taken from Rosetta Code and edited for this project. Only Part I, Program-A and Program-B submitted.

Problem #1:

The instances of Program-A must be able to write their common output file at the same time but any useful methods are forbidden.

Solution #1:

File locks are useful to solve this problem. F_SETLKW parameter of fcntl did well its job, it waits the appropriate moment by self.

Problem #2:

Inserting a new line at the middle of the file for Program-A without Standard C Library.

Solution #2:

First, I tried truncating and shifting the lines below byte-by-byte but this was too slow. Finally I did it with a buffer which has a size(x_funcs.h, MOVE_BUF_SIZE). Now It is much more faster.

Problem #3:

Deleting a line at the middle of the file for Program-B without Standard C Library.

Solution #3:

I have used the same solution with Problem #2 but reversed. I am copying the byte blocks from down to up.

Problem #4:

sleep system call in POSIX takes seconds for wait but we have to use milliseconds.

Solution #4:

I have used nanosleep system call with an helper function.

Problem #5:

Program-B must be aware of the ending of Program-A without any communication.

Solution #5:

The -t argument have to between 1 and 50 besides that there is at most four instances which using a single file. If there is not any non-empty lines in the input file of Program-Bs for a specific time(T) Program-As are ended. This time must be big enough since Program-As timer must be larger than Program-Bs. There is a defined number SAFETY_MULTIPLIER in x_funcs.h, the T equals to SAFETY_MULTIPLIER * MAX_SLEEP_TIME(50).

Problem #6:

Testing. There is not any given files.

Solution #6:

I have used random binary files generated from /dev/urandom.

Problem #7:

Program-Bs must be aware of each others status as running or closed.

Solution #7:

I have used the same method as Problem #5. When output file’s size does not change for some time, the program understands that the faster twin was already closed.