March 21st, 2018

CSE344 – System Programming - Homework #1

Submission deadline: 23:55, April 5th, 2018

The 1D Discrete Fourier Transform (DFT) is a transform with numerous everyday applications in signal processing, telecommunications, etc. In this particular context we will restrict ourselves to the case where it admits as input a finite sequence of real numbers and produces a complex sequence of the same length in return.

You are expected to develop a multi-process program consisting of 2 processes; let's call them process A and process B.

Process A's task is to produce endlessly sequences of N random real numbers and write each sequence into a line of a file X. The file X can contain at most M sequences.

Process B's task: to read endlessly the sequences in the file X (starting from the last entered line), and for each sequence:

a) delete the line where the sequence is located (i.e. replace it with an empty line)
b) calculate the DFT of the read sequence
c) print the DFT to the standard output.

You can imagine the file X as a "stack" of size M. Process A tries constantly to fill it with number sequences, and process B tries constantly to empty it by calculating the DFT of the last entered line (i.e. top of the stack).

Example of expected output on screen (yours might differ):

```
$./multiprocess_DFT -N 5 -X file.dat -M 100
Process A: i'm producing a random sequence for line 1: 1 2 3 4 1.1
Process A: i'm producing a random sequence for line 2: 2 3 8 9 1
Process B: the dft of line 2 (2 3 8 9 1) is: 23+0i -10.51-1.31i 4.01-2.12i 4.01+2.12i -10.51+1.31i
Process A: i'm producing a random sequence for line 2: 3 4 4 2 4
Process B: the dft of line 2 (3 4 4 2 4) is: 17+0i 0.61+1.17i -1.61+1.90i -1.61-1.90i 0.61+1.17i
..
..
..
```

Notes

- if the file is empty, process B must wait politely for the file to contain at least one sequence to process.
- if the file is full with M sequences, then process A must wait politely until process B has removed at least one line from it, so there'll be room for a new sequence.
- processes A and B must not modify the file X at the same time for obvious reasons.
- in case of CTRL-C make sure your processes exit gracefully by closing any and all open files.
- the parent process must clean up after its child.
- both processes **must write their every output/action on a log file of their choosing** as well.

Rules

a) You **can use only** file locks and signal related system calls for resolving any and all communication and synchronization issues between the two processes (no pipes or fifos!).
b) **No busy waiting!** And you **must not use** sleep, nanosleep or pause (you can use sigsuspend).
c) Your program must handle eventual errors gracefully according to the POSIX traditions.
d) If the command line arguments are missing or are wrong, your program must print usage information.
e) Use POSIX and Standard C libraries. You can write your program using C11 standards.
f) Valgrind rule from previous homework still holds and will be hold on every homework.

g) *Your program MUST not give "segmentation fault" or any other fault that causes your program to end in a bad way. It is better for you to send a little part of the homework with clean and executing code instead of full homework with faults like this.*
h) **Provide a makefile to compile your homework. Do not run your program inside makefile. Just compile it.**
i) Think smart and try to do the job with as short as possible and easily understandable code. Do not send homeworks with 1000 lines that can be done in 200 lines. When you find a solution to a problem, think like this: "What could be a better solution...".
j) Test your homework using the Virtual Machine given to you.
k) No late submissions.
f) **Do not send any additional files like screenshots. Just send the makefile and source files.**

Homework format:
StudentID_HW2_CSE344.tar.gz
|→ Makefile
|→ StudentID_main.c
|→ … (Any other source files)

Teaching assistant: Ahmet Soyyiğit
e-mail: asoyyigit@gtu.edu.tr

Good luck.