Gebze Technical University Computer Engineering

CSE 344 - 2019 Spring

HOMEWORK 2
REPORT

ELİF AKGÜN
1801042251

In this homework, there are two processes: the parent process P1 and its child process. They run concurrently.

P1 reads the input file(inputPath) then it gets a 2D coordinate from every couple of bytes it read. It applies least squares method to this 2D coordinates and finds a and b (y=ax+b). Then it writes this coordinates and equations to template file that it created. It repeats this calculation for every 20 bytes that read from input file. And this calculation is critical section so it is not to be interrupted by SIGINT/SIGSTOP. When P1 finishes the reading input file, sends SIGUSR1 signal to P2. When P2 receives this signal, removes the input file. P1 terminate by closing open files.

P2 forked by P1 and it reads the temlate file that created by P1. For every line it reads, it will calculate the mean absolute error (MAE), mean squared error (MSE) and root mean squared error (RMSE) between the coordinates and the estimated line. This calculation is critical section so it is not to be interrupted by SIGINT/SIGSTOP, too. Then P2 removes the line that read and writes the coordinates, estimated equations and error metrics to output file. If there is no bytes to read, P2 waits for P1 to write to the template file. After no more bytes in the template file, P2 terminate by closing files and also remove template file.

Each process catch the SIGTERM and clean it.

First of all, I set the masks and signal handlers with sigprocmask() and sigaction(). Then I create template file with mkstemp(). Then I get 2 processes with fork().

Process P1(Parent)

In this process, firstly I opened the input file denoted by inputPath and read this file. Then for each 20 bytes that I read, I calculate a and b values in the equation (y=ax+b) with *least_square* function. Due to this calculation is a critical section, I blocked the mask that include SIGINT and SIGSTOP signals. After calculation I unblocked the mask. Then I write coordinates and equations to template file with *write_coordinates_with_equation* function. Before writing I locked the file for writing, then I unlocked. For each writing to template file, P1 sends the SIGUSR2 signal to P2. This means that “There is some line to be read.”. And P1 sends SIGUSR1 to P2 to denote reading input file is finished. End of the parent process, it close the template file (P1 has closed the input path when it read.) and prints the screen how many bytes it has read as input, how many line equations it has estimated, and which signals where sent to P1 while it was in a critical section. (To indicate which signals sent in critical section, I used sigpending().) Then it frees the pointers and wait for child(to avoid zombie process) to exit.

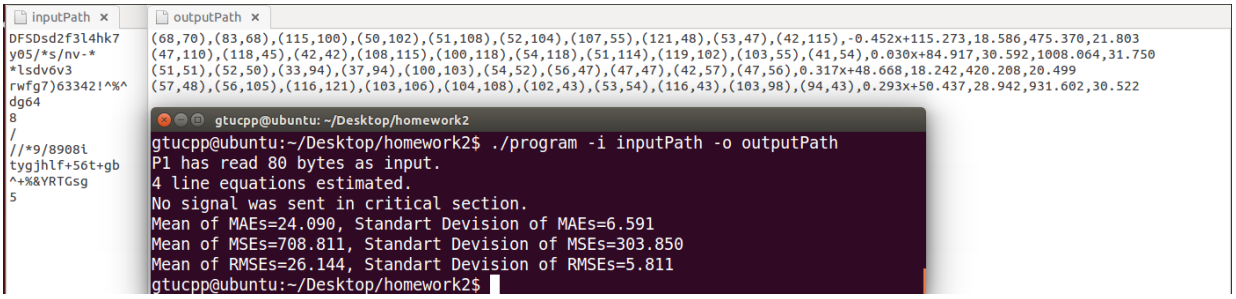
Process P2(Child)

In child process, there is a while loop. It goes on until these is no line in the template file. In loop, first of all, P2 read the template file and checks that whether template file is empty. If it is empty, it waits for P1 to write the template file. To do this I used sigsuspend(). P2 waits until SIGUSR2 received. Also I locked the template file for reading before read then I unlocked.Then I calculated the errors with *calculate_errors* function. During this calculation, I blocked the mask that include SIGSTOP and SIGINT signals, then I unblocked the mask. After these, I indicate the offsets with *where* variable to delete read line. I opened the output file denoted by outputPath, then I write this line. After writing each line, I delete read lines with lseek(). End of child process, it closes the template file and output file, and removes the template file. It prints on screen for each error metric, its mean and standard deviation, frees the pointers.

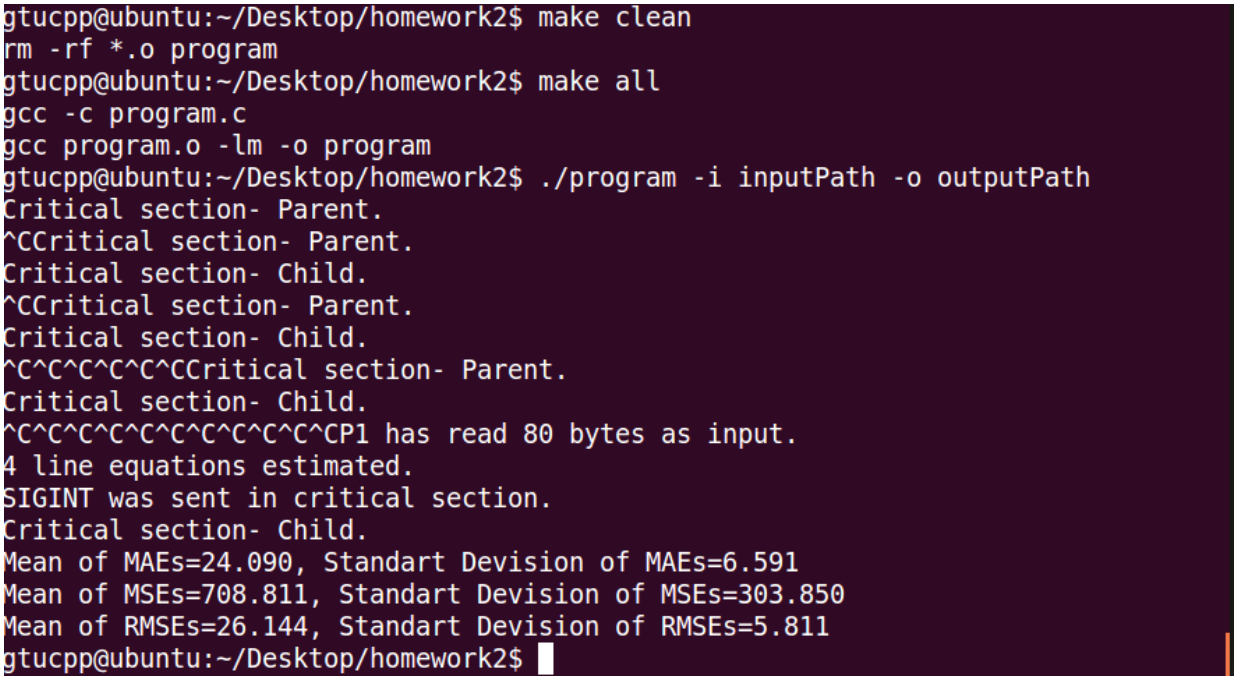
Note: Signal handler can not catch SIGSTOP signal. I wrote a handler for SIGSTOP but it does not work. And this code block causes the error so I mentioned it.

Tests:

1) Sample input and output:



2) Sending SIGINT to both process in critical section:
I send SIGINT signal to both process with Ctrl+C keys from terminal. As seen in the screen shot below, program did not interrupt. Processes catch the signal.



3) Sending SIGSTOP to parent process in critical section:
As seen in the screen shot below, program interrupted. SIGSTOP can not be catched.

```
gtucpp@ubuntu:~/Desktop/homework2$ ./program -i inputPath -o outputPath
Critical section- Parent.
^Z
[7]+  Stopped                 ./program -i inputPath -o outputPath
gtucpp@ubuntu:~/Desktop/homework2$
```

4) Sending SIGTERM to both processes
I sent the SIGTERM signal to processes with using raise(). To check whether signal is handled, I printed “SIGTERM is caught.”.

```
gtucpp@ubuntu:~/Desktop/homework2$ make clean
rm -rf *.o program
gtucpp@ubuntu:~/Desktop/homework2$ make all
gcc -c program.c
gcc program.o -lm -o program
gtucpp@ubuntu:~/Desktop/homework2$ ./program -i inputPath -o outputPath
SIGTERM is caught.
P1 has read 2600 bytes as input.
121 line equations estimated.
No signal was sent in critical section.
SIGTERM is caught.
Mean of MAEs=11.875, Standart Devision of MAEs=6.549
Mean of MSEs=294.021, Standart Devision of MSEs=218.340
Mean of RMSEs=15.361, Standart Devision of RMSEs=7.652
gtucpp@ubuntu:~/Desktop/homework2$
```

```
void handler(int signal){
  if(signal==SIGUSR1){
    usr1_interrupt = 1;
  }
  if(signal==SIGUSR2){
    usr2_interrupt=1;
  }
  if(signal==SIGTERM){
    printf("SIGTERM is caught.\n");
  }
```

5) Testing command line arguments:

As seen in the screen shot below if the command line arguments are missing/invalid my program prints usage information and exit.