GIT Department of Computer Engineering
CSE 344 - Spring 2020
Homework 1 Report

Esra EMİRLİ
151044069

1) SYSTEM REQUIREMENTS

Program A and Program B will be executed twice in arbitrary order, so there will be 4 processes running in parallel. Instances of program A will read from distinct input files, and write to a common destination file, while both instances of program B will read from program A's common destination file, modify it, and write to another destination file. All will happen at the same time.

2) PROBLEM SOLUTION APPROACH

I. Program A :

- Firstly command line arguments gets with getopt() function and checks the usage input faults.
- Then input path opens with error checks.
- Lock is defined and output path opens with controls.
- For every 32 bytes that it reads.
- writelock is wanted to be set but first checks if it has lock. If there is a pre-made lock, it is asleep until time and lock status is checked repeatedly.
- When it is able to lock, the empty line in the output path is checked. lseek goes to that line.
- Values read in input path are made as complex number and written as 16 complex numbers on the appropriate line in the output path.
- Unlocks for output path.
- The function is called to read again to read 32 bytes. If the file doesn't contain 32 more bytes then it will exit and close the file

II. Program B:

- Firstly command line arguments gets with getopt() function and checks the usage input faults.
- Then input path opens with error checks.
- Lock is defined and output path opens with controls.

- It is wanted to make a writelock so that no input is entered by Program A in input path. If there is a pre-made lock, it is asleep until time and lock status is checked repeatedly.
- A random line is selected in the input path.
- Byte corresponding to the beginning of that line is calculated.
- Check if that line is empty.
- If it is not empty, calculateFile3 function is used to read and make the necessary calculations.
- The selected line in this function is read and complex numbers are recorded.
- These numbers are calculated with the FFT algorithm and given to printResult to write the results.
- Lock is made to avoid confusion while writing to the output path. Results are written to the file. The output path is unlocked and exited.
- A space is printed on the line read in the input path.
- Output is unlocked and it searchs linearly during the cycle to check if input path is empty.
- When the input path is empty, it is closed and exited.

System Problems

1. If Program B was run without running Program A, I was getting the core dump error because there was nothing in input path(file3)

Solutions

When the Program A runs, it will write “a” in an empty file that name “esra” and each Program B will first check the existence of this file. If there is no file, it will be put to sleep to wait for Program A to run. If there is, it will increase the character in it by 1 and decrease it by 1 when exiting. The program that saw the last word "a" will delete the file. Because of

this Program A and B have to run in same folder to access “esra.txt”.

2. There are 32 characters in 1 line in file1 and file2 and if there are 20 lines in both, it writes 40 lines complex number in file3. Because:
20 * 32 = 640 bytes
20 (from newline) + 640 = 660 byte in file1 also file2.
660 / 32 = 20 lines from file1 to file3
660 / 32 = 20 lines from file2 to file3
Total 40 lines

There are 32 characters in 1 line in file1 and file2 and if there are 80 lines in both, it writes 164 lines complex number in file3. Because:
80 * 32 = 2560 bytes
80 (from newline) + 2560 = 2640 byte in file1 also file2.
2640 / 32 = 82 lines from file1 to file3
2640 / 32 = 82 lines from file2 to file3
Total 164 lines

- As a result, I perceive everything in the file (including the new line) as characters.

3. When a non-empty line is found, it’ll read those 16 complex numbers, delete that line by overwriting it with a ‘\n’. But I overwrite it with space (“ “) instead of newline(“\n”)

3) RUNNING AND RESULTS

Command line arguments :

```
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$ make
gcc -c programA.c
gcc -c programB.c
gcc programA.o -o programA
gcc programB.o -o programB -lm
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$ ./programA -i
USAGE : ./programB -i inputPathB -o outputPathB -t time
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$ ./programA -i file1 -o file3 -x
USAGE : ./programB -i inputPathB -o outputPathB -t time
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$ ./programA -i file1 -o file3 -x 8
./programA: invalid option -- 'x'
Unknown option: x
USAGE : ./programB -i inputPathB -o outputPathB -t time
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$
```

```
wustenblume@emirli: ~/Deskto
File  Edit  View  Search  Terminal  Help
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$ ./programB -i file1 -o file3 -t 51
Time range is [1,50] -t [1,50]
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$ ./programB -i file1 -o file3 -t
USAGE : ./programB -i inputPathB -o outputPathB -t time
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$
```

- Program A runs first with file1 :

file1 | programB.c | programA.c

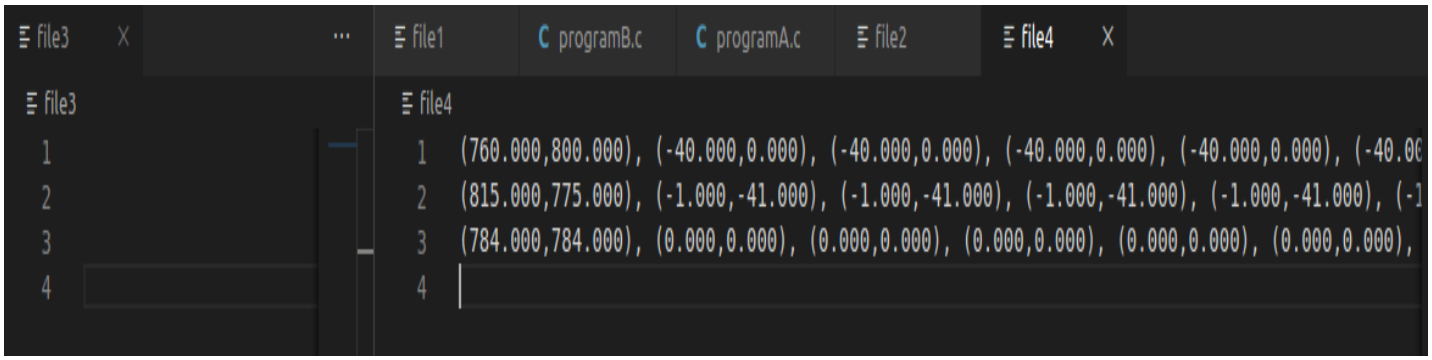
```
1  11111111111111111111111111111111
2  22222222222222222222222222222222
3  33333333333333333333333333333333
```

Result :

file1 | programB.c | programA.c | file2 | file3 | file4

```
1  49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,49+i49,
2  10+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,50+i50,
3  50+i10,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,51+i51,
4
```

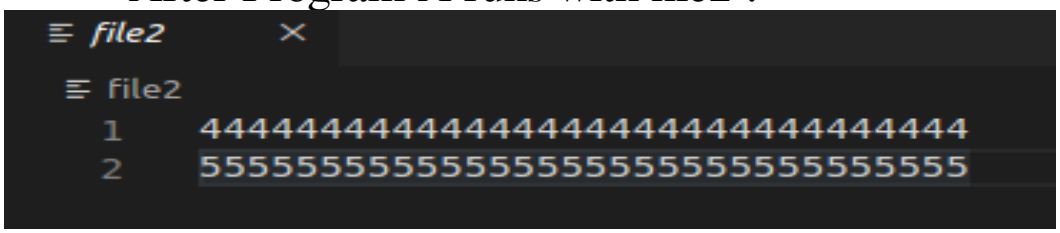
After Program B run :



■ Program B runs first :

```
gcc programA.o -o programA
gcc programB.o -o programB -lm
wustenblume@emirli:~/Desktop/SYSTEM/sistem8$ ./programB -i file3 -o file4 -t 3
There is no output from program A yet
There is no output from program A yet
There is no output from program A yet
There is no output from program A yet
There is no output from program A yet
There is no output from program A yet
There is no output from program A yet
There is no output from program A yet
```

After Program A runs with file2 :



Result :