GTU

**DEPARTMENT OF COMPUTER ENGINEERING**

**CSE 344 – Spring 2022**

**HOMEWORK 4  
REPORT**

SÜLEYMAN GÖLBOL 1801042656

# REQUIREMENTS

***NONFUNCTIONAL REQUIREMENTS***

1. Portability → The application should be portable. All computers that have Linux Distro and GCC compiler can run the program.

2. Maintainability → In case of an error occurrence, the system uses perror in order to give feedback on terminal.

3. Performance → The system should initially be able to process as many entries as possible. Each request must be processed with different terminals. The system’s performance should be fast enough to show user the feedback.

***FUNCTIONAL REQUIREMENTS***

In order to compile the program, user have to use “make” command that uses gcc. If make or gcc is not installed user can install it via *“sudo apt-get install build-essential”* command.   
Make command runs 1 command.

**gcc -Wall \*.c -o hw4**  
To run, we need to use command line arguments with parameters C, N and F.

**./hw4 -C 6 -N 2 -F files/file.txt**

If input file exists and we have permission to read the input file, the executable will run successfully.

In order to run, also C should be bigger than 4, N should be bigger than 1. Last of all, fize of file should be equal to 2\*C\*N.

1. **PROBLEM SOLUTION APPROACH**

Firstly, the first big problem for me was the using System-V semaphores with its other name semaphore sets.

While I was creating semaphore sets, I set semflag in semget() without necessary permissions so it didn’t work, then I realized and fixed. Then I created SemUnion struct to use in semctl to give necessary value.

To create threads, I had problems with understanding how concurrently works so I it wasn’t concurrent firstly, then I fixed with putting pthread\_join in other loop.

I made supplier thread detachable and consumers joinable.

Supplier thread reads byte by byte then gets the values of semaphores to print current semaphore values/amounts then posts semaphore with semctl() (with checking value of readed byte), then prints delivered values. At the end, it uses semctl() again to remove semaphore arrays.

Consumer thread reads semaphore values with semctl(), then waits for semaphore values to release. (It uses sembuf for semaphore operation, then it uses semop() to wait for both. If one is available it doesn’t get.) Then it just prints values after consuming. After N iteration, it exits.

Last problem I encountered was printing timestamp with formatted output with same print function. For this I created a function called tprintf and to merge timestampt with formatted string I used snprintf. Then with variadic function I used variadic list and vprintf to print formatted text.

**3 ) TEST CASES AND RESULTS**

# 3.a) ./hw4 -C 10 -N 6 -F files/file2.txt

# If the size and C,N values doesn’t match it print error message and exits.

# 

# 3.b) ./hw4 -C 10 -N 6 -F files/file2.txt

# When values are good, it first creates consumer threads then creates supplier threads.

# Then it goes like this when reading and consuming.

# At the end everyone exits with cleaning.

# *SEMAPHORE ARRAY AT THE END*

# 

# 

# *VALGRIND MEMORY RESULTS*

# The output from valgrind about heap and leaks is like below:

# 

# *CHECKING FOR ZOMBIE PROCESSES*

