GTU

**DEPARTMENT OF COMPUTER ENGINEERING**

**CSE 312 – Spring 2023**

**HOMEWORK 2  
REPORT**

SÜLEYMAN GÖLBOL 1801042656

# REQUIREMENTS

***NONFUNCTIONAL REQUIREMENTS***

1. Portability → The application should be portable. All computers that have Linux Ubuntu 14.04 Virtual Machine with GCC compiler can run the program.

2. Maintainability → In case of an interrupt occurrence, the system in order handles interrupt to give feedback on terminal or to use system calls.

3. Performance → The system’s performance should be fast enough to show user the feedback.

***FUNCTIONAL REQUIREMENTS***

These are the requirements that I install in order to run the program.

# sudo apt-get update

# sudo apt-get install build-essential

After installing those, I installed Oracle VirtualBox on my computer and inside the VirtaulBox, I used the VirtualBox machine which is provided by lecturer.

These are the commands that I use to create the binary file.

**make clean  
make run**

1. **PROBLEM SOLUTION APPROACH AND REQUIREMENT ACHIEVEMENT**

Binary and Linear Search -> One thread for binary, one thread for linear search.

**3 ) FUNCTIONS**

I have created different classes for the job.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Disk helps to write to file.
* Externs makes external variables available to other classes so that other classes can use them.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Info contains classes for arguments and sizes etc.
* Memory contains physical and virtual memory implementations
* Utils file doesn’t have a class inside. It contains all the functions needed to main, for example thread, second chance algorithm, page fault handling etc.

Note: I only implemented second\_chance algorithm due to time limitations.

**Get() details:**

For the function get, I used page tables and page table entries. (utils.h, line 55)

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Açıklama otomatik olarak oluşturuldu

To handle page faults, I used disk information.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

At the beginning of the program, I created page table as linked list. Then I filled all the page table entries. Then I created a file and wrote everything in virtual memory to disk.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

So, my program works like this:

If entry exists (present bit is 1), the page is in the physical memory. So, we don’t need to go to the disk to get the page.

If entry doesn’t exist, go to disk bring the page and handle the fault.

If entry doesn’t exist and also physical memory is full, apply second chance algorithm.

And if modified value is 1, I write the values to disk so data is not lost.

This is an example for how does it work (for page size=frame size = 2)

metin, diyagram, el yazısı, taslak içeren bir resim

Açıklama otomatik olarak oluşturuldu

So, If I want the value inside index 4, I make a division 4/frame size. Since frame size 2, it gives value 2. So we need to check for 2 in page table, if this page is present, we can check frame no value so we can get the value in physical memory.

**Set() function:**

Set function (utils.cpp 160) works like this.metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Açıklama otomatik olarak oluşturuldu

First it sets element in virtual memory, then it sets the page table entry value to modified, then it writes these information to file. Then, it removes the old information from physical memory so that it’s not get confused.

**Thread Handling**

To handle thread jobs I created thread\_handler() function. (utils.cpp 300)

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

I created 2 threads except than the main thread that will do summation operation at the end. ThreadInfo is for to give information to threads with void\* argument.

**Barrier**

I implemented barrier with conditional variable and mutex so that threads are waiting for each other after they finish their first job. This will help for synchronization.

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

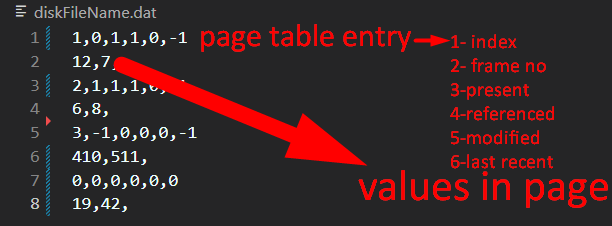
The main thread passes barrier when other 2 threads finish their first 1 job. Then this one calculates the result array and vector.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Disk File Content**

Disk contains page table entries and page values for indexes of page tables.



I couldn’t implement inverted page table structure due to time restrictions.

I couldn’t finish printing operateArrays information part.

**Running Second Chance File**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu