2020

Gebze Technical University

Department of Computer Engineering

CSE 312 / CSE 504

Operating Systems

Spring 2020

Final Exam Project

Report

Student: Nevra Gürses

NO: 161044071

Nevra Gürses
161044071

1 INTRODUCTION

1.1 Project Definition in Briefly

In this project, we will design and implement a simulated virtual memory management system and a number of page replacement algorithms in C or C++ languages. Our system will do simple integer array sorting with 4 different sorting algorithms that are bubble sort, quick sort, merge sort, index sort.We will fill the entire virtual memory with random integers .Then, we will create 4 C/C++ threads, one for each sorting algorithm. Each thread will sort 1/4 of the virtual memory in increasing order. Each thread will work on a different part of the virtual array .At the end of program,we will print our page table statistics such as number of page misses, number of accesses etc. for each algorithm as described.Our page table structure should be possible to implement Not-Recently-Used, FIFO, Second-Chance ,Least-Recently-Used , Working set clock algorithms.

2 METHOD

2.1 Page Table Structure

Before the explaining my page table structure,I will explain what is virtual memory and page table briefly.Virtual memory is that each program has its own address space, which is broken up into chunks called pages. Each page is a contiguous range of addresses. These pages are mapped onto physical memory, but not all pages have to be in physical memory at the same time to run the program.Non exist pages is keeping in disk.And so, page table is the data structure used by a virtual memory system in a computer operating system to store the mapping between virtual addresses and physical addresses. Now I will explain my page table structure in Project.

My page table structure is a struct data type that is keeping some features.I select struct data type for page table because it is keeping a number of features for page table entry and it is appropppriate for implementing Not-Recently-Used, FIFO, Second-Chance ,Least- Recently-Used , Working set clock algorithms.

If I show this structure:

```
16
17  /*Page table unit structure. */
18  typedef struct pageTableUnit{
19      int pageFrameNum; //page frame number.
20      int present; //present/absent bit.
21      int modified; //modified bit.
22      int reference;//reference bit.
23      int accessingTime; //accesssing time.
24  }pageTableUnit;
25
```

In my page table entry structure:

- **pageFrameNum:** Gives the frame number in which the current page we are looking for is present.
- **present:**Gives the information about particular page we are looking for is present or absent.If it is present it returns 1,if it is not present it returns 0.
- **modified:** Gives the information about page has been modified or not. When a page is written to, my program sets the modified bit as 1.
- **reference: :** Gives the information about page has been referenced or not.The referenced bit is set whenever a page is referenced, either for reading or for writing.
- **accessingTime:** Gives the information about last accessing time of page.I keep integer number for accessing time.If page is used I increase this integer number,it means this page used recently.This information is required for Least recently used algorithm.

My Page Table Entry Structure:

	Accessing time	Present	Referenced	Modified	Page frame number

Page Table Entry Structure Figure

2.2 VIRTUAL MEMORY AND PHYSICAL MEMORY

For simulation of virtual memory,I use an integer C array for my physical memory and virtual memory.That are:

```
24  }pageTableUnit;
25  int* virtualMemory; //virtual memory as array.
26  int* physicalMemory; //physical memory as array.
27
```

Since physical memory and virtual memory has page table entry in themselves,I make a pageTableUnit structure data type physical memory and virtual memory that are:

```
27
28  pageTableUnit* virtualMem; //virtual memory with page table unit.
29  pageTableUnit* physicalMem; //physical memory with page table unit.
30  int fifoIndex=0;
```

These are keeping each page table entry informations.All of them are above is working parallell and together.If I explain this with an example:

If frame size is given 3, physical memory frame size is given 4,and virtual memory frame size is given 5 and in commandline argument:

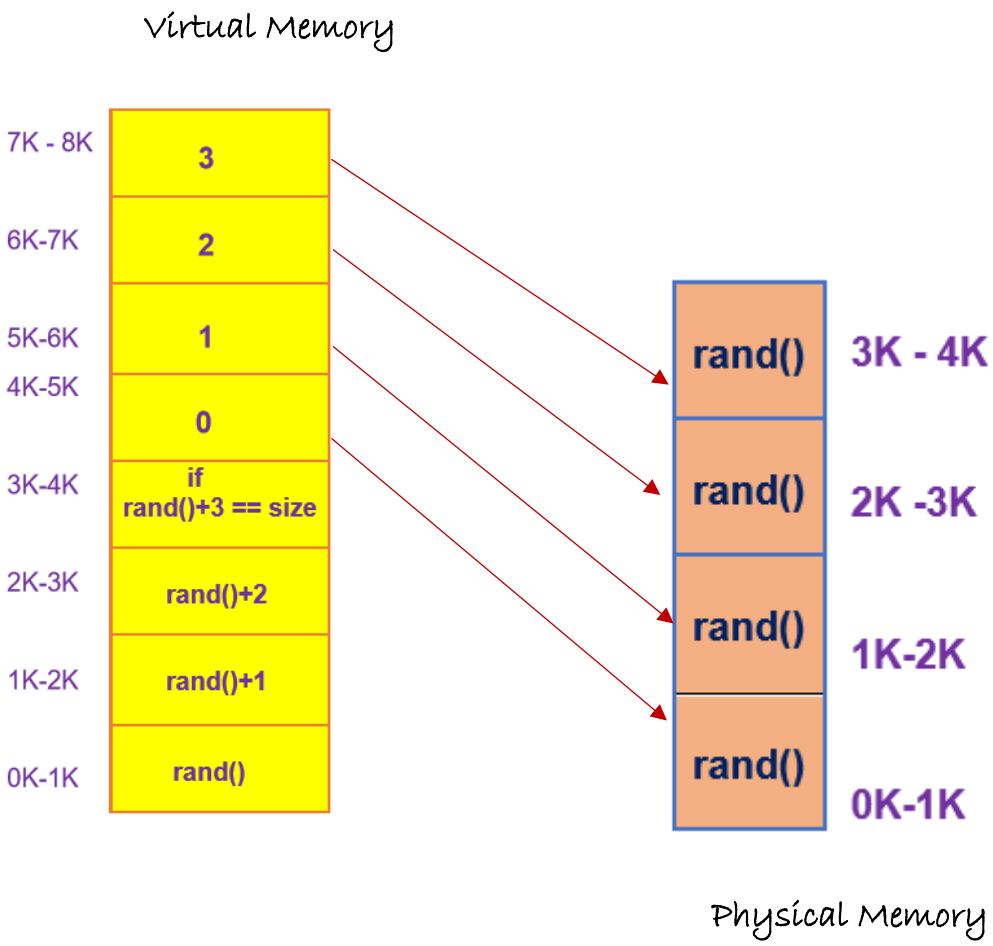
That defines frame size of 2^3 = 8 integers, 2^4= 16 physical frames, 2^5=32 virtual frames that means this system has a physical memory that can hold 8*16=128 integers and has a virtual memory that can hold 32*8= 256 integers.

So in my program virtualMemory pointer array size becomes 256 and physicalMemory pointer array size becomes 128.And VirtualMem pageTableUnit size becomes 32 and physicalMem pageTableUnit size becomes 16 and that keep page table entry informations in themselves.

Now,I will explain the mapping between virtual addresses and physical addresses.

In my system,Virtual memory keeps addresses of physical memory page frames and this physical page frames are keeping random numbers.I select a random number for physical memory address in virtual memory frame.Then I increse this random number one by one.If it achieve virtual memory size,I take this number 0 and again I increase one by one.So,I assign physical memory addresses in virtual memory.In physical memory,I take random numbers with rand function.So I create mapping between virtual memory and physical memory.For example if the address in virtual page frame is 0,this frame is mapping with 0. index of physical memory frame and so on.As expected, our virtual memory may be large. I keep our data that does not fit in your physical memory on a disk file.For example,if physical memory can keep 32 integer and virtual memory can keep 64 integer,Addresses until 32 is mapping with physical memory but other datas that are from 32 to 64 are keeping in disk file.Now,I show this structure with example figure:

I select random addresses between 0 and total virtual memory size.



Disk that includes random values of virtual memory:

Random value
Random value
Random value
Random value
Random value
Random value
Random value
Random value

In backing store that is disk file,I keep random variables of virtual memory.While searching an address in disk,it goes true line in disk and gets datas.

- An example virtual memory keeps 1024 integer in one frame.Total frame number is 8 and total integers are 8192.
- An example physical memory keeps 1024 integer in one frame.Total frame number is 4 and total integers are 4096.

2.3 REPORT OF PART-2

int get(unsigned int index, char * tName) Function:

Get function in my code that is line 422 provides that getting value in physical memory or disk.If searched address in virtual memory exist in physical memory,data is taken directly from physical memory,if it is not in physical memory page fault occurs and then data is get from disk with using appropriate page replacement algorithm.

I use helper get functions in my actual get function.These helper get functions for getting data from disk if the searched address is not in physical memory with different page replacement algoritms.If I explain used helper get functions in actual get function:

- **int getDiskWithFIFO(int increase,int newPageNum):** This function gets data from disk according to searched address in virtual memory.It implements FIFO page replacement algorithm.
- **int getDiskWithLRU(int increase,int newPageNum):** This function gets data from disk according to searched address in virtual memory.It implements LRU page replacement algorithm.
- **int getDiskWithNRU(int increase,int newPageNum):**This function gets data from disk according to searched address in virtual memory.It implements NRU page replacement algorithm.
- **int getDiskWithSC(int increase,int newPageNum):** This function gets data from disk according to searched address in virtual memory.It implements SC page replacement algorithm.
- **int getDiskWithWSClock(int increase,int newPageNum) :**This function gets data from disk according to searched address in virtual memory.It implements WSClock page replacement algorithm.

I will explain all of this page replacement algorithms after,but now I will explain getting data in physical memory currently.I write functions to make this operation.These are:

- **int controlPhysical(int pageNum):** This function for controlling whether searched frame is in physical memory or not.
- **int getPhysicalMem(int index):** This function gets found data in physical memory.

void set(unsigned int index, int value, char * tName):

This function sets given address with given value.I use this set function in fill function in mu code.

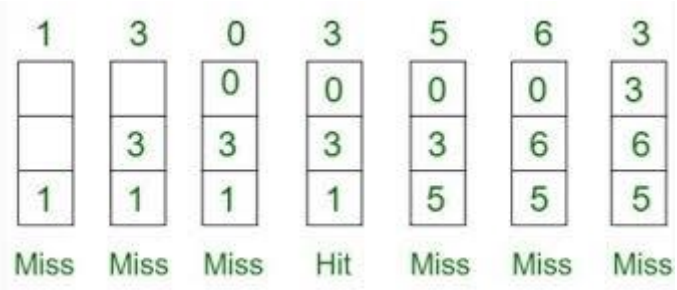
Now,I will explain page replacement algorithms:

PAGE REPLACEMENT ALGORITHMS:

❖ FIST IN FIRST OUT ALGORITHM (FIFO) :

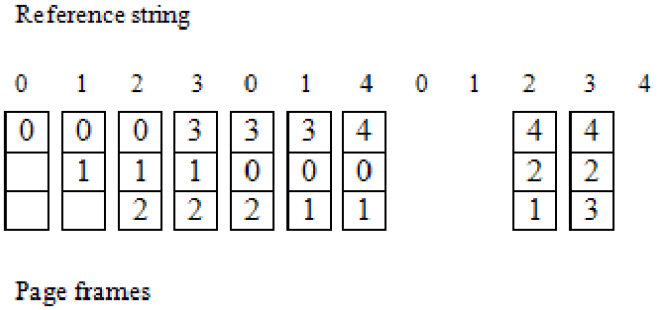
In this algorithm, my system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs that means a page fault occurs to be replaced page in the front of the queue is selected for removal.I keep a global fifoIndex for keep trace of queue.I take searched data from disk,I replace page in physical memory with get data and I increase fifoIndex.I implement fifo algorithm **int getDiskWithFIFO(int increase,int newPageNum)** in my code.

- If I show FIFO algorithm with figure:



FIFO page replacement figure 1

Or:



FIFO page replacement figure 2

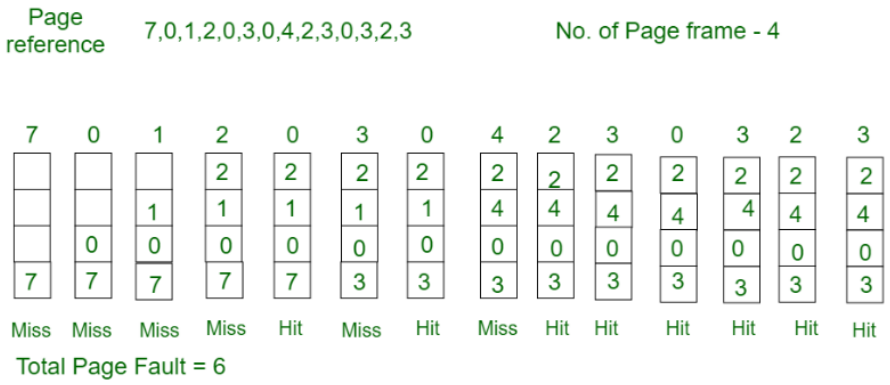
Nevra Gürses
161044071

❖ LEAST RECENTLY USED ALGORITHM (LRU) :

In this algorithm, page will be replaced which is least recently used. So,when a page fault occurs, throw out the page that has been unused for the longest time. This strategy is called LRU (Least Recently Used) paging.In my code,I keep accessingTime of pages,if a page is referenced,I increase accessing time that means that page is used newly.So my program selects which frame is replacing by determining for accessingTime.

I implement LRU algorithm **int getDiskWithLRU(int increase,int newPageNum)** in my code.

- If I show LRU algorithm with figure:



LRU page replacement figure

❖ NOT RECENTLY USED ALGORITHM (NRU) :

The not recently used (NRU) page replacement algorithm is an algorithm that favours keeping pages in memory that have been recently used. When a page fault occurs,my system inspects all the pages and divides them into four categories based on the current values of their R and M bits:

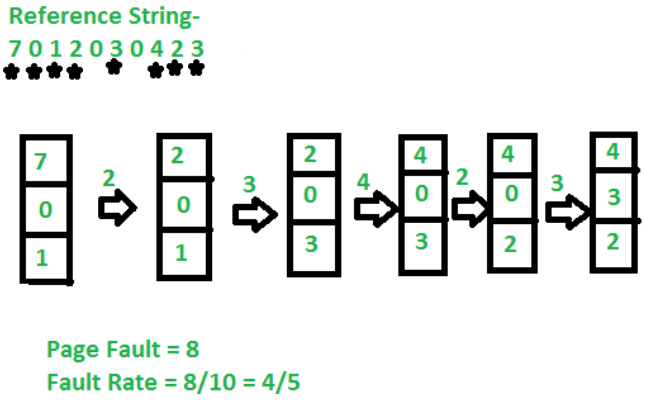
- Class 0: not referenced, not modified.
- Class 1: not referenced, modified.
- Class 2: referenced, not modified.
- Class 3: referenced, modified.

In my code,which page is replacing is determined with this classifications.Lowest classification is selected with controls.If I take a code part in my Project to show this:

```
for(i=0;i<physicalNum;i++){
    if(physicalMem[i].reference==0 && physicalMem[i].modified==0){
        fistClass=1;
        firstkeep=i;
    }
    if(physicalMem[i].reference==0 && physicalMem[i].modified==1){
        secondClass=1;
        firstkeep=i;
    }
    if(physicalMem[i].reference==1 && physicalMem[i].modified==0){
        thirdClass=1;
        firstkeep=i;
    }
    if(physicalMem[i].reference==1 && physicalMem[i].modified==1){
        fourthClass=1;
        firstkeep=i;
    }
}
```

Classification part in my code

- If I show NRU algorithm with figure:



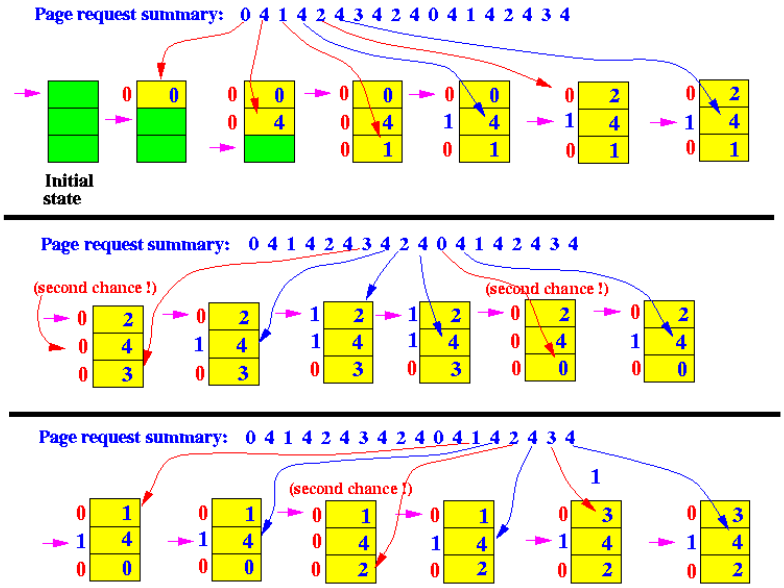
NRU page replacement figure

❖ SECOND CHANGE ALGORITHM (SC) :

This algorithm is modified version of FIFO. In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin matter, and a page that has been accessed between consecutive considerations will not be replaced. The page replaced is the one that, when considered in a round robin matter, has not been accessed since its last consideration. In implementation,adding a second chance bit to each memory frame.Each time a memory frame is referenced, setting the second chance bit to 1 this will give the frame a second chance.A new page read into a memory frame has the second chance bit set to 0.When there is need to find a page for removal, looking in a round robin manner in the memory frames:

- If the second chance bit is 1, reset its second chance bit 0 and continue.
- If the second chance bit is 0, replace the page in that memory frame.

If I show SC algorithm with figure:

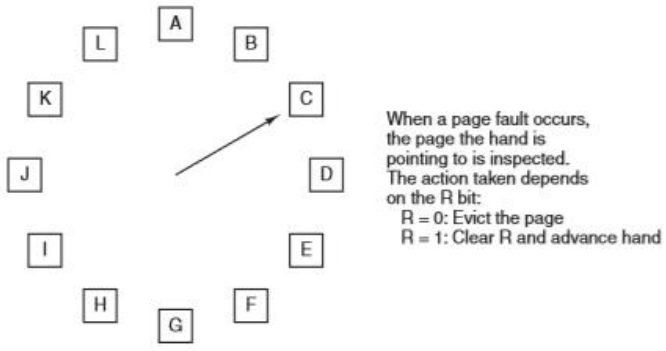


SC page replacement figure

❖ WORKING SET CLOCK PAGE REPLACEMENT ALGORITHM (WSClock) :

In that algorithm, when a page fault occurs, the page being pointed to by the hand is inspected. If its R bit is 0, the page is evicted, the new page is inserted into the clock in its place, and the hand is advanced one position. If R is 1, it is cleared and the hand is advanced to the next page. This process is repeated until a page is found with R = 0.I maket his implementation according to modified and referenced bits.

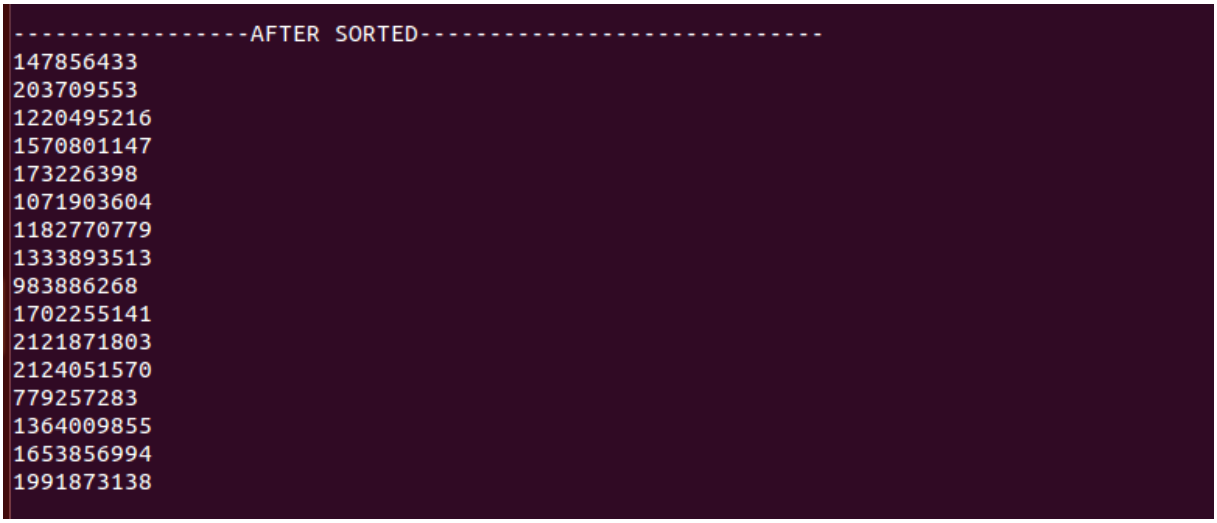
- If I show WSClock algorithm with figure:



WSClock page replacement figure

Nevra Gürses
161044071

SORTING EXAMPLE OF MY CODE:



Each frame is sorted in themselves. (For every 4 number)