

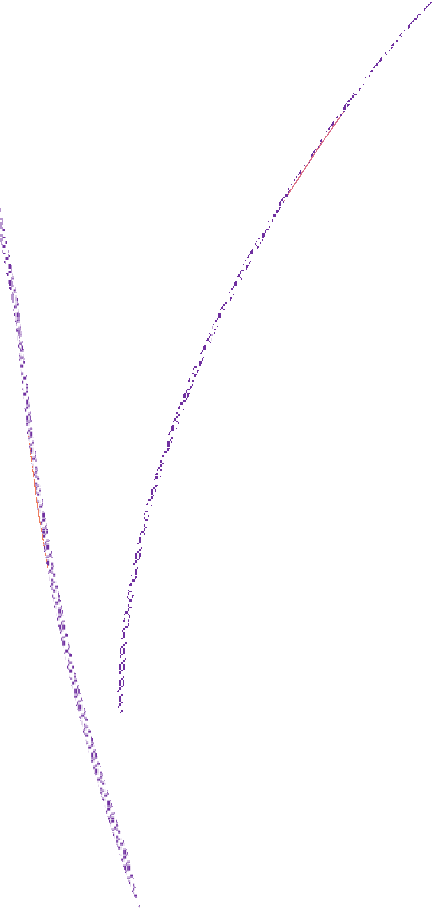
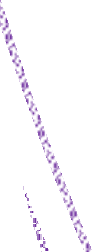
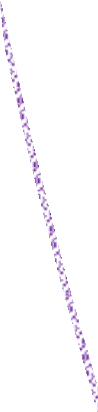
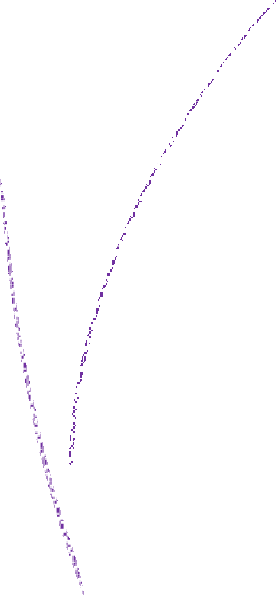
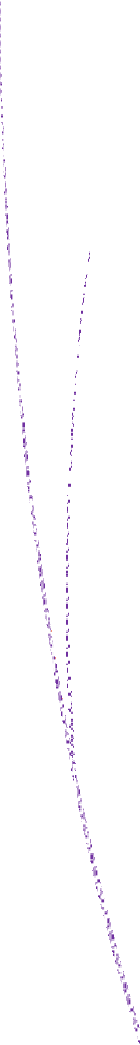
**Faculty of Engineering**

**Electrical and Computer Engineering Department**

**Operating Systems**

**ENCS3390**

**Project Report**



**“Virtual Memory Management Simulation”**

**Prepared by:**

**Loai Saad(1191582)**

**Dana Hammad(1191568)**

**Rama Abdlrahman (1191344)**

**Instructor:**

**Dr.[Ayman Hroub](https://ritaj.birzeit.edu/bzu-msgs/type?mttid=104&classid=185768)**

**Section : 4**

**Date : 5-1-2022**

Contents

[Abstract 2](#_Toc92859106)

[Introduction (theory): 3](#_Toc92859107)

[Design Philosophy: 6](#_Toc92859108)

[Program Implementation: 6](#_Toc92859109)

[Read File: 6](#_Toc92859110)

[Second Chance Clock: 7](#_Toc92859111)

[FIFO: 7](#_Toc92859112)

[Round Robin: 7](#_Toc92859113)

[Results and testing (simulations): 8](#_Toc92859114)

[File: 8](#_Toc92859115)

[Main Page: 8](#_Toc92859116)

[Browse files: 9](#_Toc92859117)

[File loaded successfully: 9](#_Toc92859118)

[Choose Algorithm: 10](#_Toc92859119)

[Second Chance clock Algorithm: 10](#_Toc92859120)

[FIFO: 11](#_Toc92859121)

[Conclusion: 12](#_Toc92859122)

[References 13](#_Toc92859123)

# Abstract

In this project we implemented page replacement algorithms, which were, Second Chance and Clock, the even ones in the algorithms list in project description (2,4) based on the team ID numbers, we also implemented fifo algorithm. To do this, we wrote a paging simulator. It read in a set of data files. The trace file also generated using a random number generator that produces random page numbers for each job. The processes are run Round Robin and the rate of page generation is fixed and the number of memory accesses for a job is proportional to its length, which is given at the start. We designed a very simple, friendly user interface, we also implemented scheduling part As a bonus.

# Introduction (theory):

Virtual memory allocation is separation of user logical memory from physical memory. It is a feature of an operating system that enables a computer to be able to compensate shortages of physical memory by transferring pages of data from random access memory to disk storage. This process is done temporarily and is designed to work as a combination of RAM and space on the hard disk. The benefits of the virtual memory are : Only part of the program needs to be loaded in memory for execution, physical address space can therefore be much smaller than logical address space, address spaces are allowed to be shared by several processes, and are allowed for more efficient process creation. In this project we use second chance and clock algorithms in replacement with RR CPU scheduling algorithm.

CPU scheduling algorithms used to choose one significant process to execute on the CPU from a group of processes (called programs if they haven’t execute before or currently created) that are ready and waiting to be selected by the scheduler dispatcher from the ready queue where they all are placed in when they are ready to be executed (this means that they are fetched or created or even finished doing input output or has finished it’s time slice in the past and did not finish its requirements).

About the second chance and clock algorithms:

FIFO replacement algorithm is the basic algorithm of second-chance replacement. When a page selected, we should check its reference bit. If it is 0(false), we replace this page; but if it is 1 (true), we give the page a second chance and move on to the next FIFO page. When a page gets a second chance, the reference bit is cleared, and arrival time reset to the current time. So, a page that is given a second chance not be replaced until all other pages replaced .Also, if a page used often enough to keep its reference bit set, it will never be replaced.

We can implement the second-chance algorithm (the clock algorithm) as a circular queue. A pointer indicates which page is to be replaced next. When a frame is needed, the pointer advances until it finds a page with false (0) reference bit. As it advances, it clears the reference bits. When a victim page found, the page replaced, and new page inserted in the circular queue in that position. You should notice that when all bits set, the pointer cycles through the whole queue, giving each page a second chance. It clears all the reference bits before choosing the next page for replacement. If all bits are set Second-chance replacement degenerates to FIFO replacement.

About RR CPU scheduling:

(RR) scheduling algorithm is a Multi-programming algorithm, it is similar to FCFS scheduling, but in (RR) scheduling, pre-emption is added which enables the system to switch between processes. In RR , time sharing used, it gives each process a time quantum that is specified before, this is the maximum time it can run on the CPU before the dispatcher pre-empts it .dispatcher interrupts every quantum to schedule next process and replace with the next process in the ready queue, and the first process is placed at the end of the ready queue waiting it turn to come again based on number of processes on the ready queue and their time quantum. a process may run for less than the time quantum (if it waits for I/O or finished for example) but not more than it. this algorithm has an overhead time which is called context switch time, where the processes are changing to perform on the CPU, this time has no benefits for the user but it is useful for the processes. This algorithm is easy to implement and it is starvation-free as all processes get a fair share of CPU.

Each process has some properties that we used in this project, which are:

1. **Completion Time** It is the time at which any process completes its execution.
2. **Turnaround Time** this mainly indicates the time Difference between completion time and arrival time. The Formula to calculate the same is:**Turn Around Time = Completion Time – Arrival Time**
3. **Waiting Time (W.T):** It indicates the time Difference between turnaround time and burst time. And is calculated as **Waiting Time = Turn Around Time – Burst Time**

.

# Design Philosophy:

We built the system using C# windows forms application. The program first reads the number of processes, size of physical memory in frames, minimum frames per process and processes information (PID, start, Duration, Size, memory traces) from the file that contains system information. Then run second chance replacement algorithm and show the results.

The simulator alters the system state by testing multiple criteria when the value of a variable represented by time increases in the program. It is verified, for example, if the presently running process has completed; if so, the completed process will be removed from the ready queue and replaced with a new process from the system queue. Another criterion to verify is whether or not pre-emption is required, which is dependent on the algorithm's pre-emption limits.

# Program Implementation:

## Read File:

Using open File Dialog, the system allows users to browse files. Then the file will open to read. Then, read first 3 lines and parse them to integer number.

After that, the system will read process lines one by one and split each line, the first index after splitting the line is process PID, second index is start time, third index is duration, forth index is size and the other are memory traces (starting from index 4).

The system then will remove 12 bits from each memory address. And check if it’s smaller than physical memory size. If the process has a smaller number of frames than the minimum, then the process will not be added.

## Second Chance Clock:

After a page has been identified, we examine its reference bit. If the value is false, we will replace this page; however, if the value is true, we will give the page a second opportunity before moving on to the next FIFO page. A page's reference bit is cleared, and its arrival time is reset to the current time when it receives a second opportunity. As a result, a second-chance page will not be changed until all other pages have been replaced (or given second chances). Furthermore, a page will never be changed if it is utilized frequently enough to maintain its reference bit set.

## FIFO:

After a page has been identified, we examine its reference bit. If the value is false, we will replace this page; however, if the value is true, it will not be replaced again.

## Round Robin:

Pre-emption is provided to allow the system to transition between processes, comparable to FCFS scheduling. A time quantum, also known as a time slice, is a tiny unit of time. A circular queue is used to treat the ready queue. The CPU scheduler iterates over the ready queue, assigning the CPU to each task for a 5 quantum time period.

# Results and testing (simulations):

## File:

**Text

Description automatically generated with medium confidence**

## Main Page:Graphical user interface, text, application Description automatically generated

## Browse files:

**Graphical user interface, text, application

Description automatically generated**

## File loaded successfully:

**Graphical user interface, application

Description automatically generated**

## Graphical user interface, text, application, email Description automatically generated Choose Algorithm:

## Table Description automatically generatedSecond Chance clock Algorithm:

## Table Description automatically generatedFIFO:

# Conclusion:

From this project, we can see and test RR scheduling algorithm, and we can see and test different page replacement algorithms( second chance and clock) , compare between them, and so know what is better for a specific purpose, since some algorithms may give less page faults than some other algorithms and other algorithms may be better for other cases. Simulating Algorithms is successful, we can see their efficiency before applying them in real. This project improved our understanding of work of CPU scheduling algorithm RR which prevent starvation , and about the replacement algorithms I mentioned above . Also we leant how to use threads in implementing . And it was very helpful and useful to have project like that , this project was very accurate in coding. We implemented an interactive GUI to simulate algorithms.

# References

Operating System Concepts, 10th Edition. . (n.d.).