BIRZEIT UNIVERSITY

Faculty of Engineering & Technology – Electrical & Computer Engineering Department

First Semester 2021 – 2022

Operating Systems– ENCS3390.

Virtual Memory Management Simulation

Name: Ali Mohammed	Name: Ahmaide Al-Awawdah
ID: 1190502	ID: 1190823
Section: 1	Section: 3
Instructor: Dr. Ahmad Afaneh	Instructor: Dr. Ayman Hroub
Date: 3rd – 9th January 2022	

1. Abstract

This is a memory management and virtual memory project where a set of data is ridden from an input file that contains the number of processes, size of physical memory, minimum frames per process, all the N processes with their id, duration time, start time, size, and all their traces.

The CPU scheduling algorithm that is used in this project is the round robin algorithm (RR) between the processes with fixed time Quantum.

For the page replacement the used algorithms are the First in First Out algorithm (FIFO), and the Least Recently Used algorithm (LRU) where the simulated machine will have a fixed number of frames to share between the running processes.

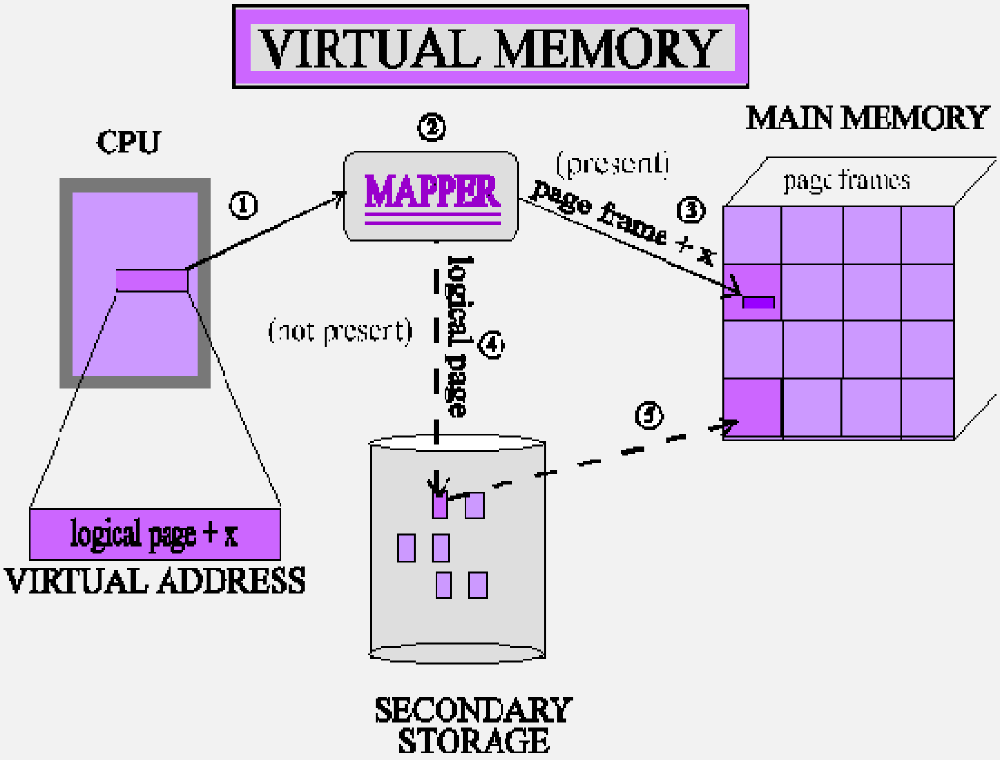


Figure 1: Virtual memory diagram

2. Theory

2.1. Process

A process in computing is the instance of the computer program that is being executed by one or more threads, and the execution must progress in sequential fashion.

2.2. Threads

A thread is a flow of execution through the process code, with its own program counter, registers that hold variables, and a stack that contains the execution history, threads are used to increase the codes efficiency. Most modern applications are multithreaded, which means that the process is executing two or more threads simultaneously for maximum utilization of the CPU as in doing multiple operations together, and without affection of other threads exception.

2.3. CPU Scheduling

The CPU scheduling algorithm in this project is the Round Robin scheduling, where each process gets a small unit of CPU time that's called time quantum where when a process finishes that time it goes all the way back to the end of the ready queue, so it goes as rotated shifting of processes.

2.4. Page Replacement

There are two-page replacement algorithms that are used in this project in order to choose which page frame to switch with and they are:

2.4.1. First in First Out

In this paging algorithm the operating system tracks all the pages in the memory in a queue where the oldest arrived page is in front of the queue and it is selected to be swapped out when a new page is coming.

2.4.2. Least Recently Used

This paging algorithm looks the previously used pages, where the ones that are heavily used are more likely to be used in the next instructions, so it swaps out the least used pages in the last few instructions with the new coming page.

3.The Program

The virtual memory simulator is designed using Java programming language. The simulator is has a user interface (UI) containing the main requirements of this project. The simulator contains multiple options which are:

- Loading file
- Generating data randomly
- Round Robin scheduling table
- First in first out data
- Least recently used data

The program hierarchy is as following:

There's a class for the process which contains all required parameters such as process id, arrival time, duration, size, and memory traces. This class which will be used from all processes.

Each page has it's class which is part of page table and frame. However, each frame got it's class which by default contains page and bit reference.

Memory has a class which contains multiple frames that are referenced during the simulation.

The core class is page replacement which contains the scheduler and page replacement polices.

These all classes are implemented together to perform as virtual memory simulator which is the target of this project. However, the main results of this simulator will be displayed on different screen windows.

3.1. The starting page of the simulator

The figure below shows the starting page that shows up, which asks the user to enter the info file name or to browse it and then to generate the data from the file by pressing on the "Generate Data" button.

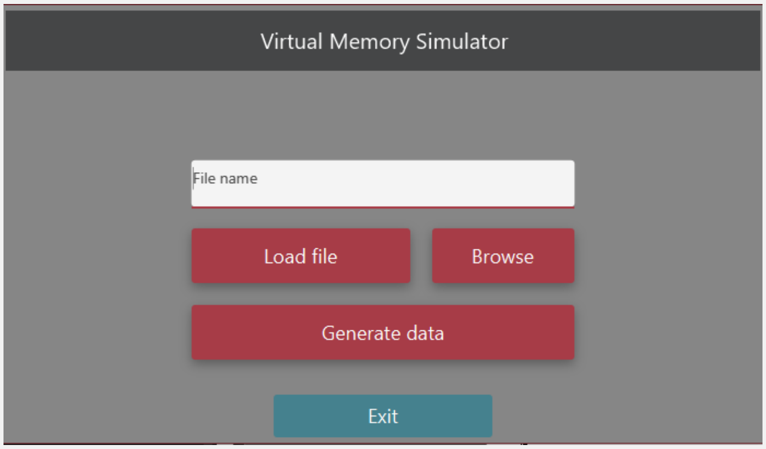
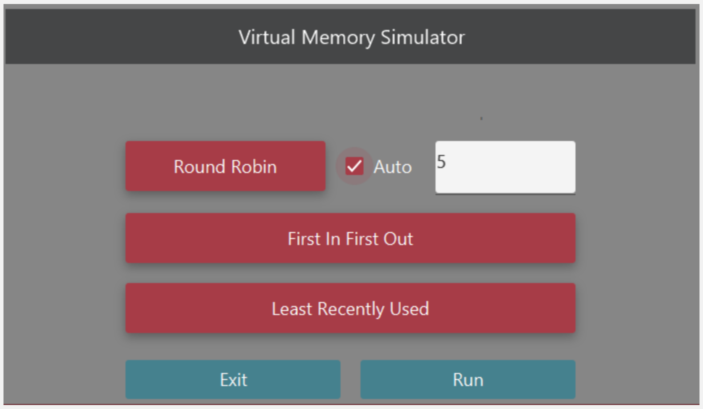


Figure 2: Starting page

3.2. The main page of the simulator

This page shows up after the user enters the file and presses on the generate data button in the starting page, as shown in the figure below in this page the user has the choice to enter the value of the time quantum for the CPU scheduling algorithm (round robin) manually or make it have its default value which is equal to 5, it also gives the user the choice to choose the page swapping algorithm (FIFO or LRU), after all that the user presses on the "Run" button in order to run the simulation.

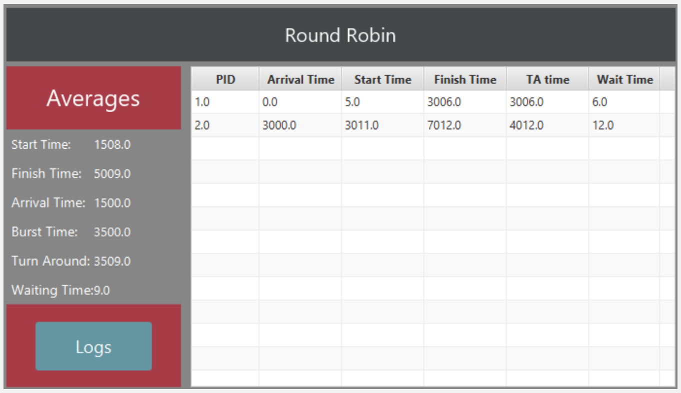


☑ Auto 5

Figure 3: Main page

3.3. Round robin page

The page in the figure below shows the info of CPU scheduled pages, as it shows for each page its, ID, arrival time, start time, finish time, turnaround time, and waiting time. This page also contains the averages for all these processes information, and it has a button to view the logs.



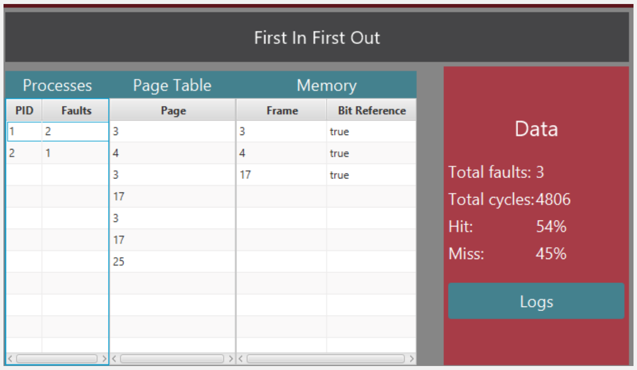
Start Time: 1508.0
Finish Time: 5009.0
Arrival Time: 1500.0
Burst Time: 3500.0
Turn Around: 3509.0
Waiting Time:9.0

PID	Arrival Time	Start Time	Finish Time	TA time	Wait Time
1.0	0.0	5.0	3006.0	3006.0	6.0
2.0	3000.0	3011.0	7012.0	4012.0	12.0

Figure 4: RR page

3.4. First in First Out page

This page replacement algorithm has a page as shown in the figure below that contains a schedule that shows all the processes with each process's ID, number of page faults, number of pages in the page table, number of frames in the memory, and the status of the bit reference weather if it was accessed before or not. This page also shows the number of total faults, and the number of total cycles, the percentage of hits and misses too.



Processes

PID	Faults
1	2
2	1

Page Table

Page
3
4
3
17
3
17
25

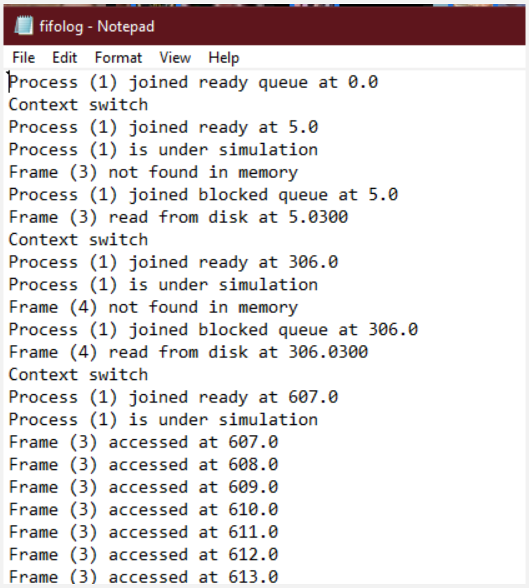
Memory

Frame	Bit Reference
3	true
4	true
17	true

Total faults: 3
Total cycles:4806
Hit: 54%
Miss: 45%

Figure 5: FIFO page

The figure below shows the logs file of the FIFO algorithm.

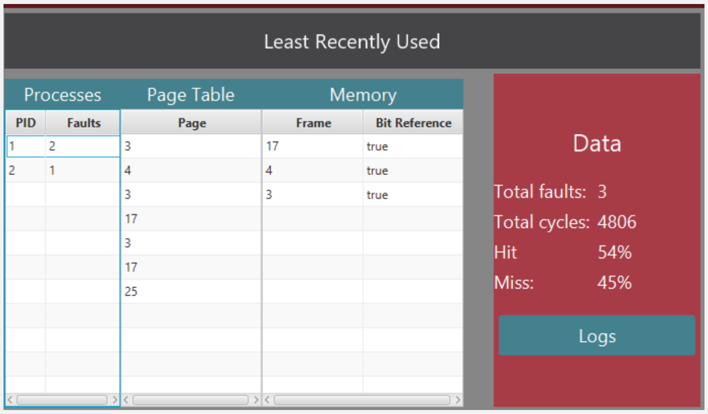


```
Process (1) joined ready queue at 0.0
Context switch
Process (1) joined ready at 5.0
Process (1) is under simulation
Frame (3) not found in memory
Process (1) joined blocked queue at 5.0
Frame (3) read from disk at 5.0300
Context switch
Process (1) joined ready at 306.0
Process (1) is under simulation
Frame (4) not found in memory
Process (1) joined blocked queue at 306.0
Frame (4) read from disk at 306.0300
Context switch
Process (1) joined ready at 607.0
Process (1) is under simulation
Frame (3) accessed at 607.0
Frame (3) accessed at 608.0
Frame (3) accessed at 609.0
Frame (3) accessed at 610.0
Frame (3) accessed at 611.0
Frame (3) accessed at 612.0
Frame (3) accessed at 613.0
```

Figure 6: FIFO logs file

3.5. Least Recently Used page

Same as the previous page, this page contains a page replacement algorithm (the LRU algorithm), where it contains a schedule for all processes with each process's ID, number of faults pages and frames, and weather if it was accessed before or not, and this page also contains the total numbers of faults and cycles, and the percentages of hits and misses.


Least Recently Used

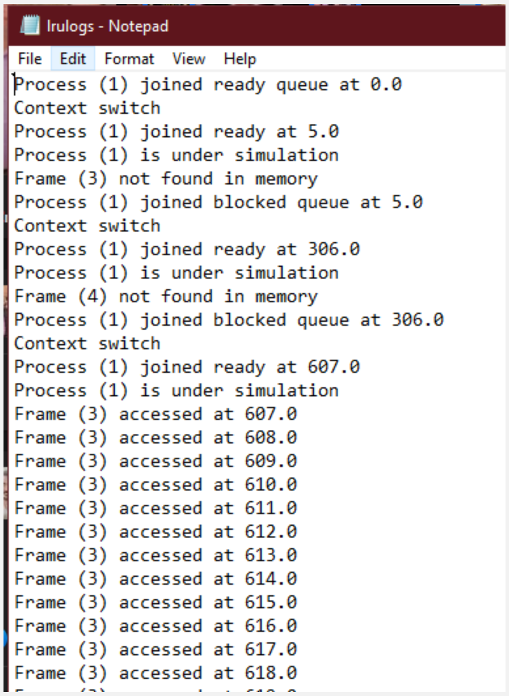
Processes		Page Table	Memory	
PID	Faults	Page	Frame	Bit Reference
1	2	3	17	true
2	1	4	4	true
		3	3	true
		17		
		3		
		17		
		25		

Data

Total faults: 3
Total cycles: 4806
Hit 54%
Miss: 45%

Figure 7: LRU page

The figure below shows the logs file of the LRU algorithm.



```
Process (1) joined ready queue at 0.0
Context switch
Process (1) joined ready at 5.0
Process (1) is under simulation
Frame (3) not found in memory
Process (1) joined blocked queue at 5.0
Context switch
Process (1) joined ready at 306.0
Process (1) is under simulation
Frame (4) not found in memory
Process (1) joined blocked queue at 306.0
Context switch
Process (1) joined ready at 607.0
Process (1) is under simulation
Frame (3) accessed at 607.0
Frame (3) accessed at 608.0
Frame (3) accessed at 609.0
Frame (3) accessed at 610.0
Frame (3) accessed at 611.0
Frame (3) accessed at 612.0
Frame (3) accessed at 613.0
Frame (3) accessed at 614.0
Frame (3) accessed at 615.0
Frame (3) accessed at 616.0
Frame (3) accessed at 617.0
Frame (3) accessed at 618.0
```

Figure 8: LRU logs file

4.Conclusion

In conclusion we learned how to create a virtual memory management simulator that reads a file data containing the memory traces of each process then simulates the required paging simulation, and how to interface it to the users where they can get their needed information easily.

We used round robin as our CPU scheduling algorithm in order to allow programs to alternate running, where the context switching takes only 5 cycles.

For the page replacement we used both FIFO and LRU replacement algorithms, we were able to compare between them and see the difference in order to see which one serves the most for the needed purpose by giving less page faults for the current case.

5. References

- Operating System Concepts Tenth Edition.

- https://www.cs.umd.edu/users/meesh/cmsc411/website/projects/virtual/virtual.html