COMP2240 A3 REPORT

C3238805 NI ZENG

The Designed java program is to simulate the Processor execution behavior with the consideration of the use of the Main memory. The designed java program can contribute to two types of memory modes both using Round Robin for I/O requests.
This program use an array to simulate the time unit. Each of the array slot represent one unit of time.

How I tested the program:

To run the program, I first compile A3.java then test it with different number of frames and different quantum of the algorithm while using the same processor's input. Notice that the faults will not change if only increase the number of total frames.

After an observation of the outcome results, I then compile A3.java again and this time use the same number of frames and same quantum but use different processor's input (eg. increase the number of processor's input) and monitor the result.

Comparison of the page replacement methods:

Fixed Allocation with Local Replacement Scope:

If the processor's allocated memory is full, the program will run a FIFO -local replacement. The program will only replace the page within its processor's allocated memory. The program will replace page using FIFO algorithm. After remove the first page which in the processor's allocated memory, the program will update the page's order in the main memory base on the page's ready Time.

On the other hand, if the main memory is not full, the program will find the next empty frame within its own allocated memory and add page into the empty frame.

Variable Allocation with Global Replacement Scope:

If the main memory is full. The program will remove the page which entered the memory the earliest then sort the memory array in FIFO algorithm. The program then can add the new page as the last element in the memory array.

However, if the main memory is not full, the program will find the next empty frame slot and add corresponding page to it.
This program will also clean up the memory space if certain processor has completed all its instructions while in Variable Allocation with Global Replacement Scope.

Interesting observations:

By Comparing "Fixed Allocation with Local Replacement Scope" and "Variable Allocation with Global Replacement Scope" output result. There is an interesting observation result in comparing the number of faults between the "Fixed Allocation with Local Replacement Scope" and "Variable Allocation with Global Replacement Scope" which having the same input condition.

Under the same input condition, "Fixed Allocation with Local Replacement" tent to have more faults if number of input processors increase while the total number of frames are small. Because

of the “Fixed Allocation” memory allocation partition frames remain the same though the entire programming process, Thus, it will often have more faults if handling large number of processors compare to “Variable Allocation with Global Replacement”.

Edge cases considered in the designed java program:

When designing this java program, there are some edge cases that are considered (List below):

If user enter only the number of frames and quantum but no processor input, the program will show a message to the user and exit the program.

Each input processor allow to have Maximum 50 pages, if input more than 50 pages of a processor, the program will displace an error messages and terminate the program.

Trick/technique

Draw down a draft of what the question is asking and analyze it before coding really help me to understand the questions deeper and have a basic idea on how to construct the coding program. By drawing down how and when the processor needs to be into a queue actually help me to find out if I have any logical issues with my code.

Common out each classes properly and have them organized does always helps me to debug easier.