THE UNIVERSITY OF NEWCASTLE AUSTRALIA

**School of Information and Physical Sciences**
**Semester 2, 2022 - COMP2240/COMP6240 - Operating Systems**

# Assignment 3 (15%)

Submit using Canvas by **11:59 pm, Friday 21st October 2022**

## 1. Task:

In assignment 1, we assumed that the system had an infinite amount of memory. In this assignment, the operating system has a limited amount of memory and this needs to be managed to meet process demands. You will write a program to simulate a system that uses paging with virtual memory. The characteristics of the system are described below:

- **Memory:** The system has **F** frames available in user memory space. During execution, the processor will determine if the page required for the currently running process is in main memory.
    a. If the page is in main memory, the processor will access the instruction and continue.
    b. If the page is not in main memory, the processor will issue a *page fault* and *block* the process until the page has been transferred to main memory.
    c. Initially no page is in the memory. I.e. the simulation will be strictly *demand paging*, where pages are only brought into main memory when they are requested.
    d. In fixed allocation scheme frames are equally divided among processes, additional frames remain unused. In variable allocation scheme all frames are available to the processes.
- **Paging and virtual memory:** The system uses paging (without segmentation).
    a. Each process can have a maximum 50 pages where each page contains a single instruction.
    b. For page replacement you will need to apply ***Generalised Clock*** (***GCLOCK***) policy (described below).
    c. You will need to implement two different schemes for resident set management
        i. '***Fixed Allocation with Local Replacement Scope***' – In this scheme, the frames allocated to a process do not change over the simulation period i.e. even after a process finishes, the allocated frames do not change. And the page replacements (if necessary) will occur within the frames allocated to a process using GCLOCK policy.
        ii. '***Variable Allocation with Global Replacement Scope***' – In this scheme, no specific frame is allocated to any process rather all frames are available to the processes for use. A process can use an unused frame in the user memory space to bring in its own page. For page replacement it will use GCLOCK policy but will consider all the pages in the user memory space. I.e. the page selected for replacement may belong to any process running in the system. When a process finish execution the frames to that finished process are not returned immediately rather will be replaced by pages from other processes using GCLOCK replacement policy.
    d. All pages are read-only, so no page needs to be written back to disk.

- **Page fault handling:** When a page fault occurs, the interrupt routine will handle the fault by blocking that process and placing an I/O request. The I/O controller will process the I/O request and bring the page into main memory. This may require replacing a page in main memory using a page replacement policy (GCLOCK with local or global scope). Other ready processes should be scheduled to run while such an I/O operation is occurring.
    a. Issuing a page fault and blocking a process takes no time. <u>So multiple page faults may occur and then another ready process can run immediately at the same time unit.</u>
    b. Swapping in a page takes 6 units of time (if a page required by a process is not in main memory, the process must be put into its blocked state until the required page is available).
    c. If a process is unblocked (i.e. the requested page is placed in the main memory) at time t then it can be scheduled and the requested page can be executed at t.
- **Scheduling:** The system is to use a Round Robin short-term scheduling algorithm with time a quantum of **Q**.
    a. Executing a single instruction (i.e. a page) takes 1 unit of time.
    b. Switching the processes does not take any time.
    c. All the processes start execution at time *t*=0. And they will be processed in the order the process names appear in the input.
    d. If a process becomes ready at time unit *t* then execution of that process may occur in the same time unit *t* without any delay (if there is no other process running or waiting in the ready queue).
    e. If multiple process becomes ready at the same time then they will enter the ready queue in the order they became blocked.
    f. If a process P1 is finishes its time quantum at t1 and another process P2 becomes unblocked at the same time t1, then the unblocked process P2 is added in the ready queue first and the time-quantum expired process P1 is added after that.

You are to compare the performance of the ***GCLOCK with Fixed Allocation & Local Page Replacement*** and ***GCLOCK with Variable Allocation & Global Page Replacement*** algorithms.

Please use the basic versions of the policies introduced in the lectures.

**Generalised Clock (GCLOCK)** policy is the generalised version of the Clock policy introduced in the lecture. The frames are considered as circular buffer and initially the Clock pointer is set to indicate the first frame (as in the simple Clock policy). Each frame has a reference counter (instead of a reference bit as in simple Clock) initialized to zero. When a page is loaded the reference counter of that frames is set to zero and after the first reference to that page the reference counter is set to 1 and the clock pointer is incremented to point to the next frame. Whenever the page is subsequently referenced the reference counter is incremented by one. The replacement procedure for GCLOCK works similar to Clock policy. For replacement the GCLOCK scans through the buffer from the pointer’s current position looking for a reference counter with value zero. If the counter is not zero it is decremented by one and the pointer moves to the next frame. When a frame with reference counter zero is found its current page is replaced with the new page and the pointer is set to the next frame.

## 2. Other Submission Requirements:

Your submission must also conform to the follow considerations.

### 2.1. Programming Language:

The programming language is Java, versioned as per the University Lab Environment (*Note this appears to have been* ***rolled back on the Lab machines****, which means we're* ***currently targeting a subversion of Java 11 – so please be conscious of this***). You may only use standard Java libraries as part of your submission.

### 2.2. User Interface:

The output should be printed to the console, and strictly following the output samples given in the assignment package. While there are no marks allocated specifically for the Output Format, there will be a deduction when the result format varies from those provided.

### 2.3. Input and Output

Input to your program should be via **command line** arguments, where the arguments are system configurations and the names of files that specify the execution trace of different processes. All processes start execution at time 0, and are entered into the system in the order of the command line arguments (with the third argument being the first process). For example:

```
java A3 30 3 process1.txt process2.txt process3.txt
```

where 30 is the number of frames (**F**) and 3 is the quantum size (**Q**) for Round Robin algorithm and the other arguments are text file names containing page references for each process. These are **relative filenames**, and **SHOULD NOT BE HARDCODED** in anyway and number of processes may vary.

Since we assume that each page contains a single instruction, an execution trace is simply a page request sequence.

For example: (`process1.txt`)

```
begin
1
3
2
1
4
3
end
```

This specifies a process that first reads page 1, then page 3, and so on until the '`end`' is reached.

For each replacement strategy, the simulation should produce a summary showing, for each process, the turnaround time, the total number of page faults and the time the page faults were generated.

Sample inputs/outputs are provided. Working of the first example is presented with details of different levels for visualisation. Your submission will be tested with the above data and will also be tested with other input files.

Your program should output to standard output (*this means output to the Console*). Output should be **strictly** in the shown output format (*see the supplied output files).*

**If output is not generated in the required format then your program will be considered incorrect.**

## 2.4. Mark Distribution:

Mark distribution can be found in the assignment feedback document (`Assign3Feedback.pdf`);

## 2.5. Deliverable:

1. Your submission will contain your program source code with documentation and the report (*below*) in the root of the submission. These files will be zipped and submitted in an archive named **`c9876543A3.zip`** (*where* **`c9876543`** *is your student number*) – do not submit a `.rar`, or a `.7z`, or etc.
2. Your main class should be **`A3.java`**, your program will compile with the command line **`javac A3.java`** and your program will be executed by running **`java A3 30 3 data1.txt data2.txt data3.txt`** (where **`data?.txt`** can be any relative filename and can be any number of files; see Input and Output above for description – do not hard code anything!).

   **Note:** If your program can not be compiled and executed in the expected manner (*above*) then please add a `readme.txt` (containing any special instructions required to compile and run the source code) in the root of the submission. Please note that such non-standard submissions will be **marked with heavy penalty**.
3. Brief 1 page (A4) **report** of how you tested your program and a comparison of the page replacement methods based on the results from your program and any interesting observations. Specifically, your report should include discussion on edge cases you considered and behaviour of your algorithm on those cases, any specific trick/technique you applied and did you face any specific issue.

**NOTES**

**1.** Assignments submitted after the deadline (**11:59 pm 21st October 2022**) will have deducted 10% of the maximum marks possible, per 24 hours late, in line with UoN Policy. This means (for example) if you submit two days late, and score 80% in the assignment, you mark will be 80 (mark) – 20 (2 * 10% maximum possible mark (100%)) = 60.

**2.** If your assignment is not prepared and submitted following above instructions then you will lose most of your marks despite your algorithms being correct.