TSN2101 Operating System : Assignment Questions

Instructions:

Form a group with a maximum of **3 members** (you are allowed to form a group with less, minimum is 1 – you'll be doing the assignment alone but must in same tutorial group). **You can choose to do either Topic 1 or Topic 2 or Topic 3.**
You are allowed to use any programming language to simulate these concepts, not limited to C++, Java, Python, etc. Submit your program online on MMLS (under Assignment Report->Tutorial Session->student Name) by **28th Sept 2020, 4pm**. You are to **present** your work to your respective lecturers during the tutorials in Week 14.
Any form of plagiarism will not be tolerated. No marks will be given for this assignment if plagiarism is detected.

Topic 1:

Simulation of CPU scheduling algorithms

Process Scheduling algorithms: Choose THREE (3) scheduling algorithms

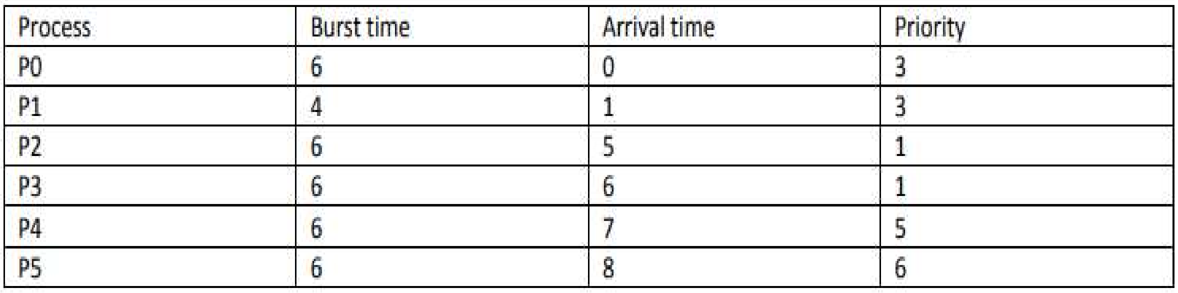
a) Round Robin with Quantum 3

b) Preemptive SJF

c) Non Preemptive SJF

d) Preemptive Priority

e) Non Preemptive Priority



Process	Burst time	Arrival time	Priority
P0	6	0	3
P1	4	1	3
P2	6	5	1
P3	6	6	1
P4	6	7	5
P5	6	8	6

Pre-assigned case

1. User should be able to enter the details about the processes such as Arrival Time, Burst Time, Priority, Time Quantum for Round Robin assigned at the beginning of simulation and the number of processes can range from 3 to 10.

2. Executing the program should show the Gantt chart (visual form) of each algorithm.

3. Calculation of

a) Turnaround time for each process

b) Total and Average Turnaround time for the entire processes

c) Average Waiting time for each process

d) Total and Average Waiting time for the entire processes

Expected display:
1. User should be able to enter the details about the processes such as Arrival Time, Burst Time, Priority, Time Quantum assigned at the beginning of simulation and the number of processes can range from **3 to 10**.

2. Executing the program should show the **Gantt chart** (text form) of each algorithm.

3. Sample inputs and outputs:

Process	Burst time	Arrival time	Priority
P0	6	0	3
P1	4	1	3
P2	6	5	1
P3	6	6	1
P4	6	7	5
P5	6	8	6

Q1: FCFS with priority

P0	P2	P3	P0	P1	P4	P5
0	5	11	17	18	22	28 34

Q2: RR with priority (quantum = 2)

P0	P1	P0	P2	P3	P2	P3	P2	P3	P1	P0	P4	P5
0	2	4	6	8	10	12	14	16	18	20	24	30 34

Q3: 3 level queue (quantum = 2)

Queue 1

▨	P2	P3	P2	P3	P2	P3	▨
	5	7	9	11	13	15	17

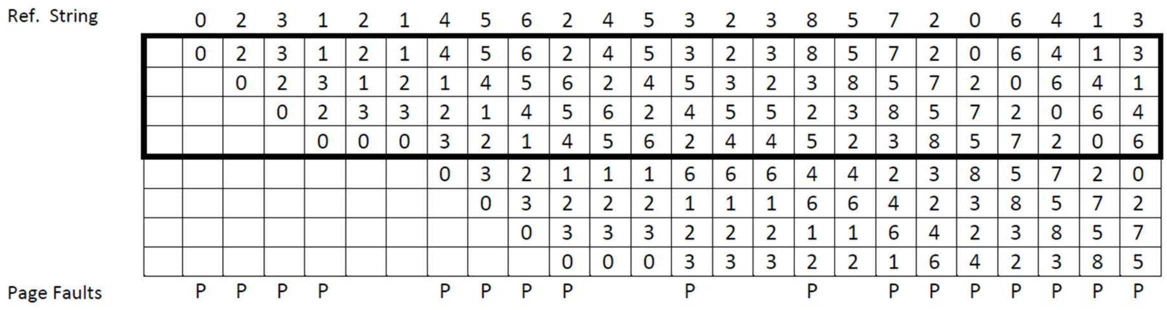
Topic 2:

Topic 2: Simulation of Virtual Memory Paging Algorithms
A paging system can be characterized by three items:
1. The reference string of the executing process
2. The page replacement algorithm
3. The number of page frames available in memory, m

Conceptually, we can imagine an abstract interpreter that works as follows. It maintains an internal array, M, that keeps track of the state of memory. It has many elements as the process has virtual pages, which we will call n. The array M is divided into 2 parts.
The top part with m entries contains all the pages that are in memory.
The bottom part, n-m pages, contains all the pages that have been referenced once but have been paged out and are not currently in memory. Initially, M is the empty set since no pages have been reference and no pages are in memory.
As execution begins, the process begins emitting the pages in the reference string one at a time. As each one comes out, the interpreter checks to see if the page is in memory (i.e., in the top part of M). If it is not, a page fault occurs. If there is an empty slot in memory (ie the top part of M contains fewer than m entries), the page is loaded and entered in the top part of M. This situation arises only at the start of execution. If memory is full (i.e., the top part of M contains m entries), the page replacement algorithm is invoked to remove a page from memory.
As an example, refer to the following example using the LRU page replacement. The virtual address space has eight pages and the physical memory has four page frames. At the top of the following figure, we have a reference string consisting of 24 pages.

0 2 3 1 2 1 4 5 6 2 4 5 3 2 3 8 5 7 2 0 6 4 1 3

Under the reference string, we have 25 columns of 8 items each. The first column, which is empty, reflects the state of M before execution begins. Each successive column shows M after one page has been emitted by the reference and processed by the paging algorithm. The heavy outline denotes the top of M, that is, the first four slots, which correspond to page frames in memory. Pages inside the heavy box are in memory and pages below it have been paged out to disk.



Ref. String		0	2	3	1	2	1	4	5	6	2	4	5	3	2	3	8	5	7	2	0	6	4	1	3
		0	2	3	1	2	1	4	5	6	2	4	5	3	2	3	8	5	7	2	0	6	4	1	3
			0	2	3	1	2	1	4	5	6	2	4	5	3	2	3	8	5	7	2	0	6	4	1
				0	2	3	3	2	1	4	5	6	2	4	5	5	2	3	8	5	7	2	0	6	4
					0	0	0	3	2	1	4	5	6	2	4	4	5	2	3	8	5	7	2	0	6
								0	3	2	1	1	1	6	6	6	4	4	2	3	8	5	7	2	0
									0	3	2	2	2	1	1	1	6	6	4	2	3	8	5	7	2
										0	3	3	3	2	2	2	1	1	6	4	2	3	8	5	7
											0	0	0	3	3	3	2	2	1	6	4	2	3	8	5
Page Faults		P	P	P	P			P	P	P	P			P			P		P	P	P	P	P	P	P

The first page in the reference string is 0, so it is entered in the top of memory, as shown in the second column. The second page is 2, so it is entered at the top of the third column. This action causes 0 to move down. In this example a newly loaded page is always entered at the top and everything else moved down as needed.
Each of the first seven pages in the reference string causes a page fault. The first four can be handled without removing a page but starting with the reference to page 5, loading a new page requires removing an old page.
The second reference to page 3 does not cause a page fault, because 3 is already in memory. Nevertheless the interpreter removes it from where it was and puts it on the top as shown. The process continues for a while until page 5 is referenced. This page is moved from bottom part of M to the top part (i.e. it is loaded into memory from disk). Whenever a page is referenced that is not within the heavy box, a page fault occurs as indicated by P's below the matrix.

The model in which this replacement algorithm is based on the following:
1. When a page is referenced, it is always moved to the top entry in M.
2. If the page referenced was already in M, all pages above it move down one position. A transition from within the box to outside of it corresponds to a page being evicted from memory.
3. Pages that were below the referenced page are not moved. (In this manner, the contents of M exactly represent the contents of the LRU algorithm)

Algorithms:

1. First-in First-Out (FIFO): A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen.
2. Least Recently Used (LRU): Replace the page that has not been used for the longest period of time.
3. Not Recently Used (NRU): An approximation to LRU, select one of the pages that has not been used recently (by keeping one bit called the 'reference bit', where 1 → used recently and 0 → not used recently). After a certain number of clock intervals, all reference bits are reset.

- Second Chance (Clock): The clock algorithm keeps a circular list of pages in memory (FIFO), with the iterator pointing to the last examined page frame in the list. When a page fault occurs and no empty frames exist, then the reference (R) bit is inspected at the hand's location. If R is 0, the new page is put in place of the page the iterator points to, otherwise the R bit is cleared. Then, the clock hand is incremented and the process is repeated until a page is replaced.
- Not Frequently Used: The not frequently used (NFU) page replacement algorithm requires a counter, and every page has one counter of its own which is initially set to 0. At each clock interval, all pages that have been referenced within that interval will have their counter incremented by 1. In effect, the counters keep track of how frequently a page has been used. Thus, the page with the lowest counter can be swapped out when necessary.
- Aging: A variation of the NFU, but the counter values are decreased over time. The reference counter on a page is first shifted right (divided by 2), before adding the referenced bit to the left of that binary number.

Assignment Requirements:

Using the model above, write a program that simulates a paging system. At the start of the program, the user input the reference string for the **NRU**, **Clock**, **NFU** and **Aging** page replacement algorithms. Each algorithm is to run the same reference string. The user will input the number of frames, from 3 to 7. For **NRU**, the user is also required to input the number of turns before the reference bits are reset. Concurrently with the processing of each of the page using the algorithms, display the where the page faults occur with respect to the different algorithms.

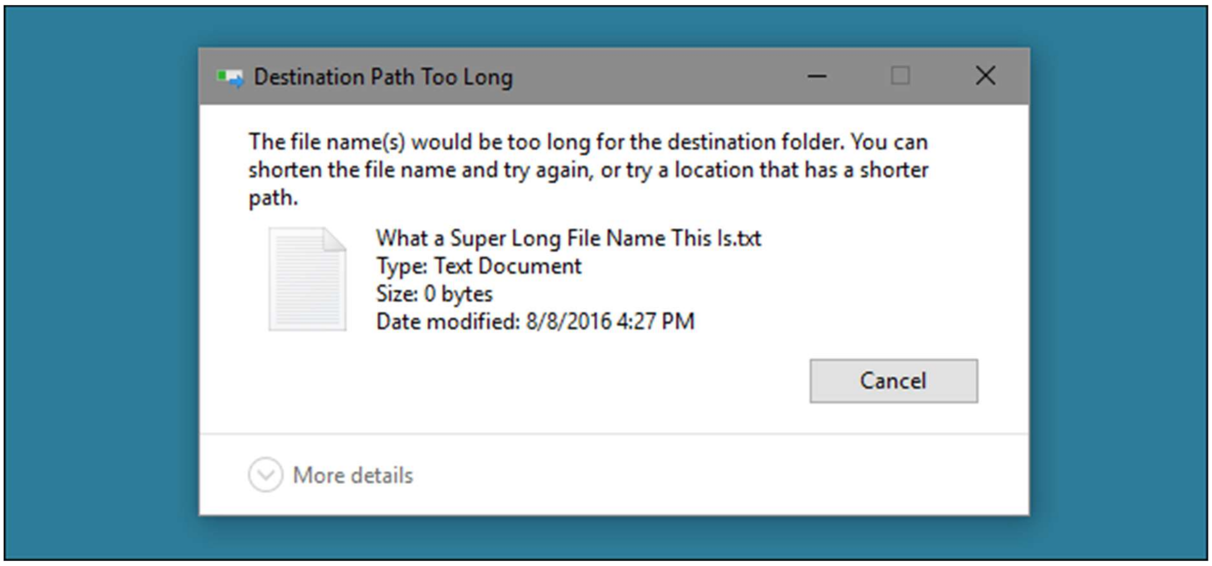
Expected display:
1. Upon running the system, the user can input the reference string and number of frames.
2. Executing the algorithm means being able to display the parallel execution for all 4 algorithms for the same string.

3. Occurrences of page faults are indicated at their respective frames.

Topic 3: Operating System Assignment

Each group need to select 3 questions from the 6 questions below.

Write a program/shell scripts that will give a filename with N characters with no extension (e.g. .exe, .com, .dll). The file is just a text file with the content "This is OS hacking test". Start N with 1, then slowing increase the value of N until the OS system cannot take it and prompt out a message similar to this. Run it on Linux, or different version of windows and get us the result.


Destination Path Too Long

The file name(s) would be too long for the destination folder. You can shorten the file name and try again, or try a location that has a shorter path.

What a Super Long File Name This Is.txt
Type: Text Document
Size: 0 bytes
Date modified: 8/8/2016 4:27 PM

1. Write a program/shell scripts that will give a filename with 3 characters but with N characters extension. The file is just a text file with the content "This is OS hacking test". Start N with 1, then slowing increase the value of N until the OS system cannot take it and prompt out a message similar to this. Run it on Linux, or different version of windows and get us the result. Example, filename.1, filename.12, filename.123, … filename123….N (you can repeat the 0123456789).

2. Write a program/shell scripts that will test the max directory structure in OS. The program will continuously make sub-directory inside a sub-directory until the OS system limit is broken (until the OS disallowed you to create further sub-directory). The first subdirectory is 1, the N subdirectory is N. Thus, c:\1\2\3\...\N.
   Run this on Various operating system and get us the result.

3. Write a program/shell scripts that will test the max number of files inside a directory c:\HackerTest. The program will continuously make duplicate of a same 1 byte size file to a different filename. Do it until the OS disallowed you to create any further file. The

first filename is file1, the last filename is fileN where N is a decimal digit. E.g N = 1234.

4. Write a recursive program will continue to call on itself until the hangs. (run out of memory). Print the value of N. The recursive program will be a recursive adding program where on the first call = 1, 2nd call = 1 + previous call. This will test the OS limit on stack.

5. Write a program/shell scripts that will continuously make duplicate of a same 10MB size file to a different filename, until your computer run out of space. Please do this with care!

Extra Notes:

Some of these program is testing the limit in your OS. Observe how the OS started to slow down and fail. Record the error message or the scenario when the OS started to slow down and fail. Best record it in video and upload it to youtube. If you can get high view count and comment, additional marks can be considered.