Santa Clara University

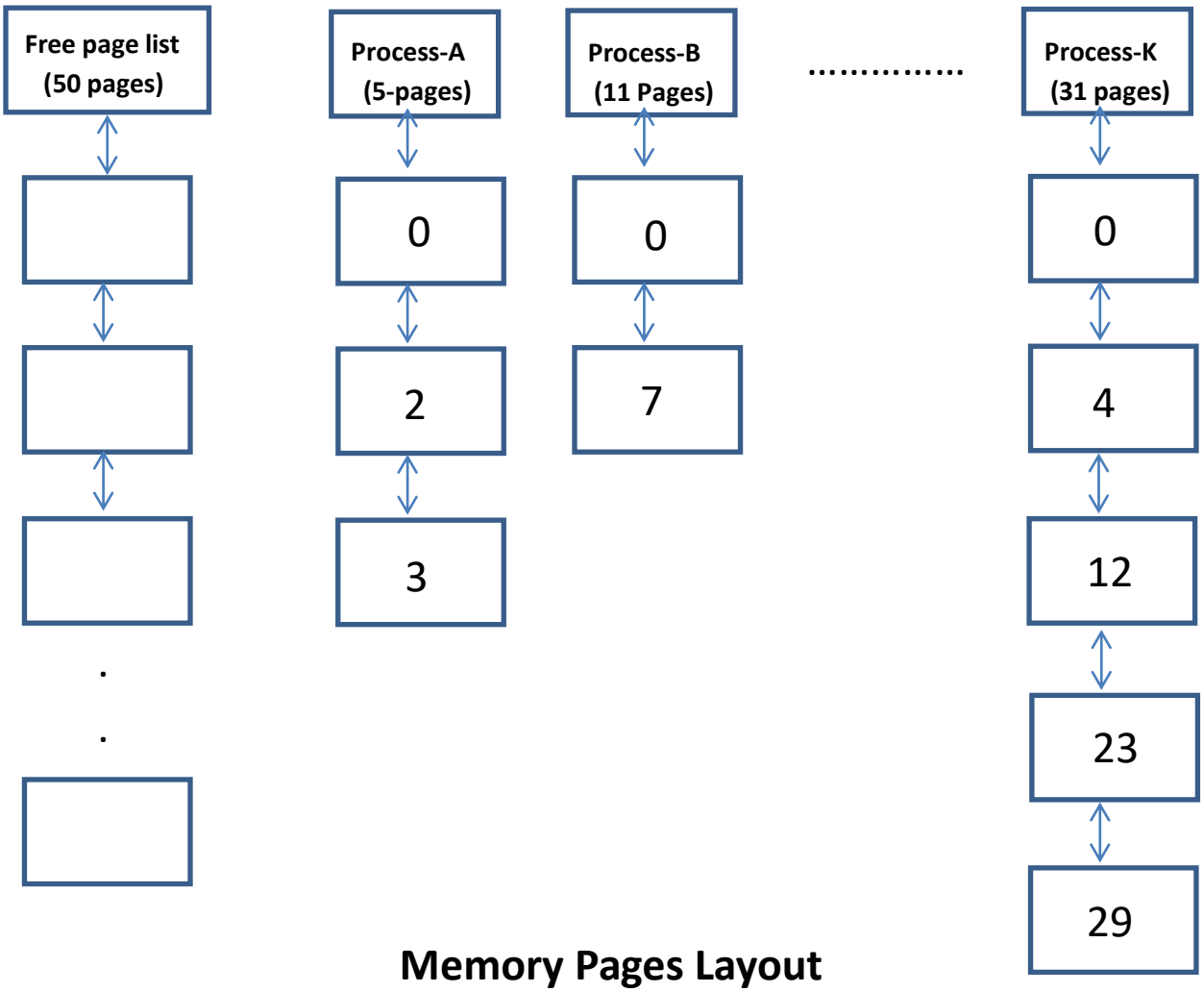
Department of Computer Engineering

Advanced Operating Systems (COEN 383)

Project-4 Preview (6 pts)

Instructor: Ahmed Ezzat

Page Replacement Algorithms Simulator



Memory Pages Layout

We will build simulation written in Java, C or C++ programming language that experiment with multiple processes running concurrently, each process do start at page-0 then every **100 msec** it references random page from its own address space

taking into consideration the locality of reference algorithm as described in the Homework assignment.

P.S. L**ocality of reference**, after referencing a page i, there is a 70% probability that the next reference will be to page i, i-1, or i+1. i wraps around from 10 to 0. In other words, there is a 70% probability that for a given i, Δi will be -1, 0, or +1. Otherwise, $|\Delta i| > 1$.

Workload Generation

We will generate 150 jobs <process Name, Process size in pages, Arrival time, Service Duration>. Sort the random jobs generation based on arrival time and having them structured as linked list. Processes have randomly and evenly distributed sizes of **5, 11, 17, and 31** MB. Processes have randomly and evenly distributed service durations of **1, 2, 3, 4, or 5** seconds.

Simulator:

1. Generate the workload and represent it as sorted queue based on arrival time
2. Create and initialize the free page list, initially with 100 pages, each is 1 MB.
3. Pick up one job at a time from the Job queue and if there are 4 free pages in the free page list then start running that process, otherwise wait till one of the existing processes complete. Each process is represented by a header and linked list of its memory resident pages.
4. Generate the appropriate record whenever starting or completing a job <time stamp, process name, Enter/exit, Size in Pages, Service Duration, Memory-map>.
5. Once a job start execution, it generates a memory reference every 100 msec to a random page from its own virtual address space; need to generate an appropriate record <time-stamp in seconds, process Name, page-referenced, Page-in-memory, which process/page number will be evicted if needed>.

6. If memory is all used and process reference a page that is not currently in memory then we need to apply the chosen "page replacement Algorithm" to select a victim page to evict so you can bring to memory the needed page.
7. Run the simulator 5 times, each is 1 minute, and each time using different replacement algorithm {algorithms **FIFO**, **LRU**, **LFU** (Least Frequently Used), **MFU (**Most Frequently Used**)**, and **random pick**}.
8. Continue running until the 1 minute expires, collect and save the requested statistics and exits.
9. Run simulator 5 times, each to complete the one minutes, and compute the hit/miss ratio of pages referenced by the running jobs for each run. Then get average of 5 runs.
10. Run the simulator for 100 page references, and for each reference, print the <time-stamp in seconds, process Name, page-referenced, if-Page-in-memory, which process/page number will be evicted if needed>.
11. For each replacement algorithm, print the average number of processes (over the 5 runs) that were successfully swapped-in.