

COSC 3360/6310 - Operating Systems Fall 2017

Programming Assignment 3 - Virtual Memory Manager with Paged Segments

Due Date: Saturday, December 2, 2017, 11:59pm CST

In this assignment, you will design a virtual memory manager with paged segments, and then simulate its operation for six different page replacement algorithms (2 were discussed in class; 4 are new or variants of those discussed in class for you to explore): FIFO (first-in-first-out), LIFO (last-in-first-out), LRU-X (least recently used), LDF (Longest Distance First), OPT-X (optimum with lookahead of X future addresses), and WS (Working Set). There will be 6 sets of output from these 6 runs.

LIFO replaces the page which came into the main memory last.

LRU-X replaces the page whose X-th most recent access is furthest in the past. For example, LRU-1 is simply LRU whereas LRU-2 replaces pages according to the time of their penultimate access. LRU-X improves greatly on LRU with regard to locality in time.

LDF replaces the page which is most distant from the current page. If two pages are at the same distance from the current one, then replace the page which is next to the current page in counter-clockwise rotation. The basic idea behind LDF is that the locality of reference is as in LRU but the difference is that in LDF, the locality is based on distance and not on the used references.

OPT-X lookaheads X page references instead of all the way to the end of the input reference string. Obviously, this approach is only possible in this simulation where all page references are known in advance.

The information provided here is intended to guide you in your design; it is not a complete description of all issues that must be considered. The assignment requires a combination of Unix/Linux processes (such as the page fault handler, disk driver, and page replacement process) synchronized by Unix/Linux semaphores, and simulators (such as the paging disk).

Data Structures:

You may add additional fields to these data structures if needed in your design. There is one segment table per address space (process); one page table per segment; and one table entry per page. The address can thus be divided into 3 parts: segment number, page number within the segment, and displacement (offset).

A single frames table contains the address space and page number of the page currently occupying each page frame. The format of the entry for frame f is:

```
FRAMES[f]:  a p forward_link backward_link
```

Entries in the frames table are linked together in allocated and free lists. The disk page table entry, DPT[a,p], contains the address on the paging disk of page p of address space a.

Key Elements of the Assignment:

1. Write a page fault handler process that can be invoked by the interrupt dispatcher when a page fault occurs. The address space and page number of the missing page are made available

to the fault handler by the addressing hardware. The fault handler requests a page transfer by placing an entry in the disk driver work queue and signaling the associated semaphore.

2. Design a disk driver process which schedules all I/O to the paging disk. The disk command

```
STARTDISK(read/write, memory_addr, disk_addr)
```

initiates an I/O operation. The paging disk is wired to the semaphore of the disk driver process, which it signals when a command is complete. The disk has a work queue containing entries of the form

```
(process_id, read/write, frame_index, disk_addr).
```

3. Assume that the page replacement algorithm runs as a separate process which is signaled each time a page frame is removed from the pool. the replacement process attempts to maintain a free pool size between min and max frames. To accomplish this, one or more processes may need to be "deactivated" and "activated" later.

Note: In this assignment, you need not consider the actual creation and deletion of address spaces. Be sure to initialize the semaphores you use with correct initial counts.

Input and Output:

The input to your simulator is as follows:

```
tp    /* total_number_of_page_frames (in main memory) */
sl    /* maximum segment length (in number of pages) */
ps    /* page size (in number of bytes) */
r     /* number_of_page_frames_per_process for FIFO, LIFO, LRU-X, LDF, and OPT-X,
        or delta (window size) for the Working Set algorithm */
X     /* used by LRU-X and OPT-X, 0 for others (which do not use this value) */
min
max
k     /* total number of processes */
pid1  size1 /* process_id followed by total number of page frames on disk */
pid2  size2
:      :
:      :
pidk  sizek
```

These parameters are followed by a list of process id and address pairs: pid addr. You need to extract the segment number and page number from the address addr. The last address accessed by process i is followed by: i -1.

the output from your simulator includes the following data. You may also show other useful statistics.

```
number_of_page_faults for each process;
for Working Set, show the min and max size of the set and
total_number_of_page_faults
```