Embedded Operating System

Paging

Page Fault

- Each page table entry contains frame address, permissions, dirty bit, valid bit, etc.
- If page is present in main memory its page table entry is valid (valid bit = 1).
- If page is not present in main memory, its page table entry is not valid (valid bit = 0).
- This (Invalid entry) is possible due to one of the following reasons:
  - Page address is not valid (dangling pointer).
  - Page is on disk/swapped out.
  - Page is not yet allocated.
- If CPU requests a page that is not present in main memory (i.e. page table entry valid bit=0), then "page fault" occurs.
- Then OS's page fault exception handler is invoked, which handles page faults as follows:
  1. Check virtual address due to which page fault occured. If it is not valid (i.e. dangling pointer), terminate the process. (Validity fault).
  2. Check if read-write operation is permitted on the address. If not, terminate the process. (Protection fault).
  3. If virtual address is valid (i.e. page is swapped out), then locate one empty frame in the RAM.
  4. If page is on disk, load the page in that frame.
  5. Update page table entry i.e. add new frame address and valid bit = 1 into PTE.
  6. Restart the instruction for which page fault occurred.

Page Replacement Algorithms

- While handling page fault if no empty frame found (step 3), then some page of any process need to be swapped out. This page is called as "victim" page.
- The algorithm used to decide the victim page is called as "page replacement algorithm".
- There are three important page replacement algorithms.
  - FIFO
  - Optimal
  - LRU

FIFO

- The page brought in memory first, will be swapped out first.
- Sometimes in this algorithm, if number of frames are increased, number of page faults also increase. This abnormal behaviour is called as "Belady's Anomaly".

OPTIMAL

- The page not required in near future is swapped out.
- This algorithm gives minimum number of page faults.
- This algorithm is not practically implementable.

LRU

- The page which not used for longer duration will be swapped out.
- This algorithm is used in most OS like Linux, Windows, ...
- LRU mechanism is implemented using "stack based approach" or "counter based approach".
- This makes algorithm implementation slower i.e. time complexity O(n).
- Approximate LRU algorithm close to LRU, however is much faster.

Global vs Local page replacement

- Local page replacement: If there is shortage of RAM, then swap-out a page of the current process itself.
- Global page replacement: If there is shortage of RAM, then swap-out a page from any process.

Thrashing

- If number of programs are running in comparatively smaller RAM, a lot of system time will be spent into page swapping (paging) activity.
- Due to this overall system performance is reduced.
- The problem can be solved by increasing RAM size in the machine.

Major vs Minor Page Faults

- If page fault handling doesn't need any disk IO, then it is minor page fault.

- If page fault handling needs disk IO (swapping), then it is major page fault. This needs more time.

Virtual pages vs Logical pages

- By default all pages of user space process can be swapped out/in. This may change physical address of the page. All such pages whose physical address may change are referred as "Virtual pages".
- Few kernel pages are never swapped out. So their physical address remains same forever. All such pages whose physical address will not change are referred as "Logical pages".
- A process may call mlock() or mlockall() syscall that prevent swapping out the pages. This in turn make pages logical. Refer manual of mlock().

```
mlockall(MCL_CURRENT | MCL_FUTURE);
```

- The page stealing process runs in background and swap out the non active pages (i.e. not in current active set) to make room for the new allocations done by the running/active processes.

Assignment

1. Simulate FIFO page replacement algorithm. Input page sequence and display page fault count.
2. Simulate Optimal page replacement algorithm. Input page sequence and display page fault count.
3. Simulate LRU page replacement algorithm. Input page sequence and display page fault count.
4. Simulate Second-chance LRU page replacement algorithm. Input page sequence and display page fault count.