Virtual Memory

Md. Asifur Rahman
Lecturer
Department of CSE, RUET

Virtual Memory

- **Virtual Memory** is a memory allocation scheme that involves separation of user perception of the logical memory from physical memory. This separation allows an extremely large virtual memory to be provided to the programmer when only a small physical memory is available.
- **Virtual Address** is assigned to location of virtual memory in order to allow that location to be accessed as though it were part of main memory.
- **Address Space** refers to the range of memory addresses available to process.

Implementation of Virtual Memory

Demand Paging

- Load pages only if they are needed for execution.
- It is similar to a paging system with swapping where processes resides in secondary memory.
- During execution instead of swapping the entire process into memory it uses a **lazy swapper** (here the term **lazy pager** would be rather correct since in this context **swapper** term is technically incorrect as swapper manipulates entire process whereas **pager** is concerned with individual pages of a process) that only swaps a page into memory if it is needed.

Demand Paging

1. When a process needs to be swapped in for execution, the **pager** guesses which pages will be used before the process is swapped out again. Thereby the pager swaps in only those pages instead of swapping the whole process.
2. The **valid-invalid bit** scheme can be used in the **page table** to mark whether pages associated with the process are on memory or are on the disk. If the **valid-invalid bit** is set to:
    - **i. Valid**: It means associate page is both legal and in main memory.
    - **ii. Invalid**: It means associate page is either not legal or is legal but not present in main memory.
3. If certain reference to a page is made during execution whose valid-invalid bit in the page table is marked as invalid then it causes page fault.

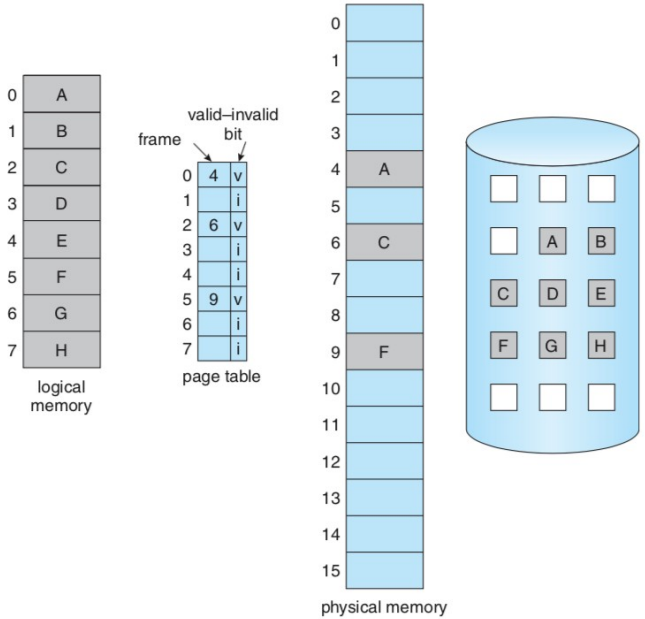

Figure 9.5 Page table when some pages are not in main memory.

Demand Paging

4. Page fault will cause a trap to OS in order to indicate OS's failure in bringing the desired page into memory.
5. Procedure involving handling page fault are:
    I. Check the internal table (usually kept with PCB) of the process to determine if the reference was a legal or illegal memory access.
    II. If the reference is illegal then the process is terminated. But if it is legal but the referred page have not yet brought in, then it needs to be paged in.
    III. To page in the referred page first a free frame is located.(Free space available or Page replacement)
    IV. Next a disk read operation of the desired page into the newly allocated frame is made
    V. After completion of the disk read operation, the page table is modified/updated to reflect the changes.
    VI. Restart the instruction that was interrupted by the trap.

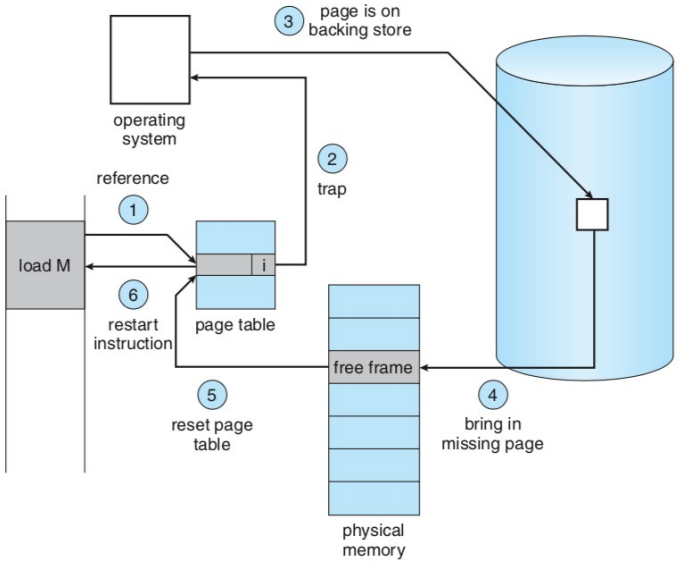

Figure 9.6 Steps in handling a page fault.

Demand Paging

- **Pure Demand Paging:** Never brings a page into memory until it is required.

- **Performance of Demand Paging:**

  Effective access time = (1-p) x ma + p x page fault service time.
  where , p = probability of page fault
  ma = memory access time
  Page fault service time majorly consists of:

  - Page fault interrupt service time
  - Page read in time
  - Process restart time

Page Replacement

Page replacement is required when no frame is free to page in the desired page from the back store or disk. It involves:

1. Locating a frame in memory which is not currently in use and then freeing it by paging out its content.
2. Change the page table.
3. Finally page in the desired page into the freed frame.
4. Reset the page table.

Page Fault Service Routine With Page Replacement

1. Find the location of the desired page on disk.
2. Find a free frame
   a. If there is a free frame, use it
   b. If there is no free frame, use a page-replacement algorithm to select a victim frame.
   c. Write the victim frame to the disk, change the page and frame tables accordingly.
3. Read the desired page into the newly freed frame, change the page and frame table accordingly.
4. Continue to user process from where the page fault occurred.

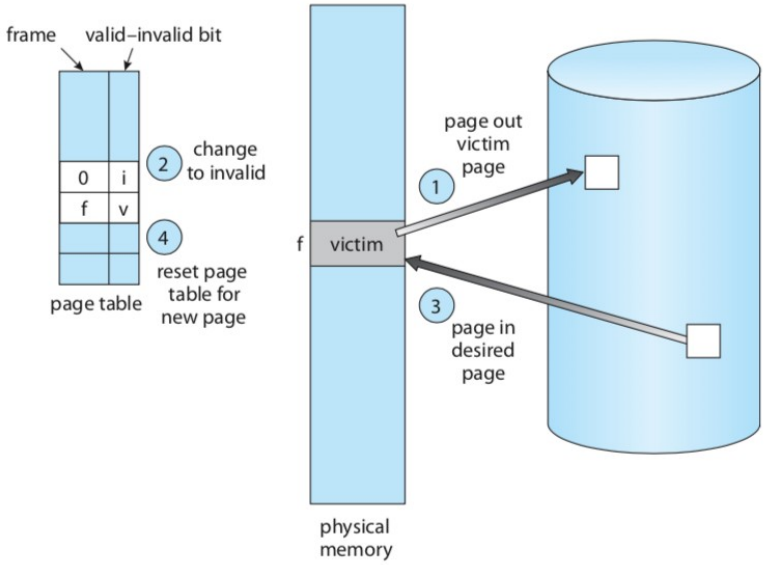

Figure 9.10 Page replacement.

Page Replacement Algorithm

- **FIFO Page Replacement:** This algorithm assign a time to each page when it was brought into memory. When a page needs to be replaced the oldest page is chosen. Instead of recording the time a FIFO queue can also be used.

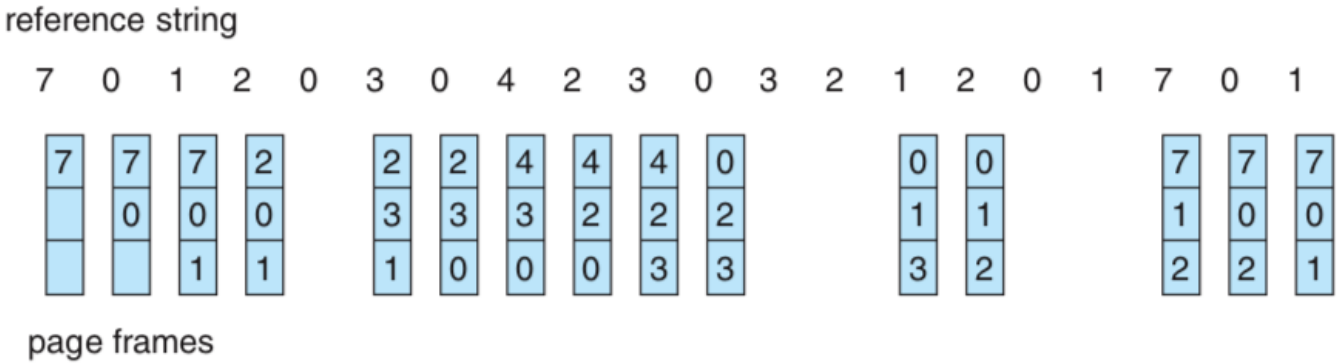

Figure 9.12 FIFO page-replacement algorithm.

Self Study advantage and disadvantages

Page Replacement Algorithm

- **Optimal Page Replacement:** This algorithm replaces the page that will not be used for the longest period of time.

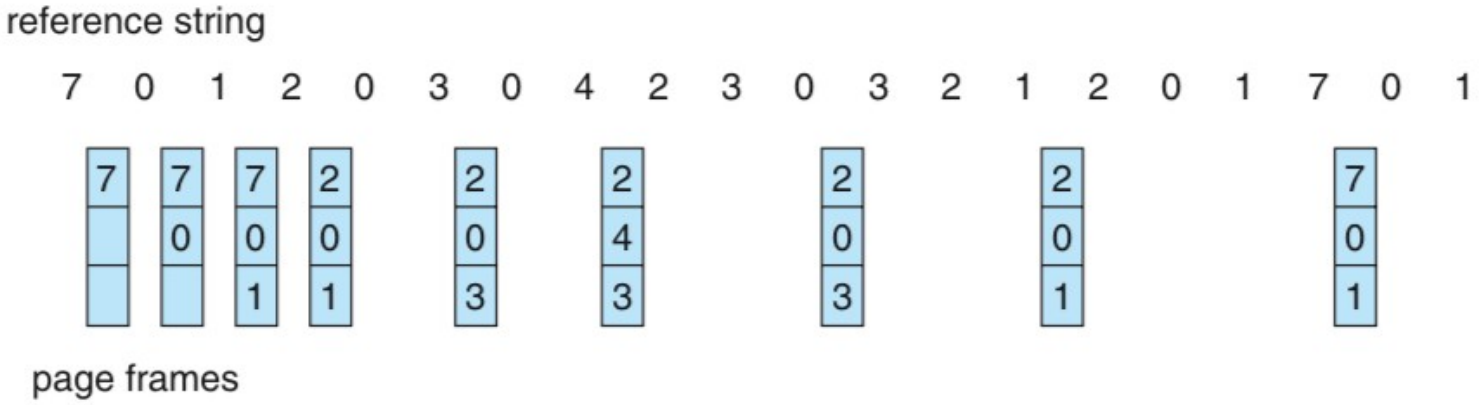

Figure 9.14 Optimal page-replacement algorithm.

Self Study advantage and disadvantages

Page Replacement Algorithm

- **LRU Page Replacement:** This algorithm replaces the page that has not been used for the longest period of time.

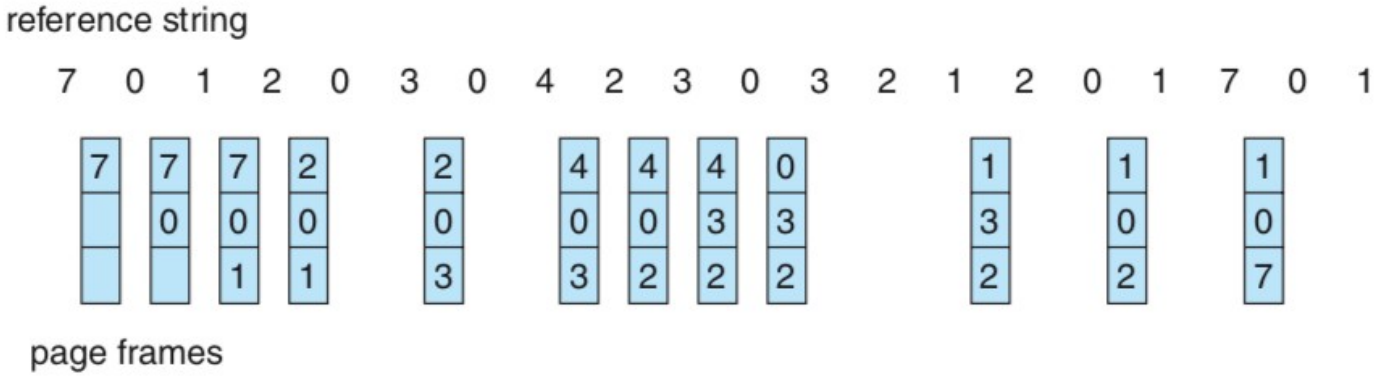

Figure 9.15 LRU page-replacement algorithm.

Self Study advantage and disadvantages

Copy on Write

- When child process in created from parent process using fork() system call, both parent and child share the same pages using **copy-on-write** scheme
- Copy-on-write scheme means if either parent or child wishes to write to a shared page, a copy of the shared page is created and mapped to the address space of the process that is trying to write to it. Hence the process trying to write on the shared page will now write to the copy of the shared page while preserving the original one.

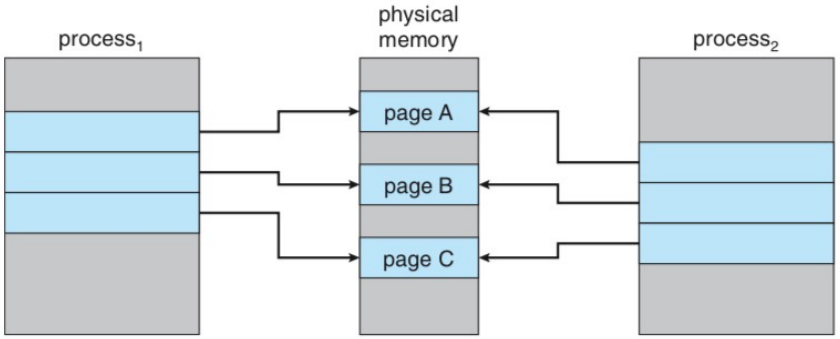

Figure 9.7 Before process 1 modifies page C.

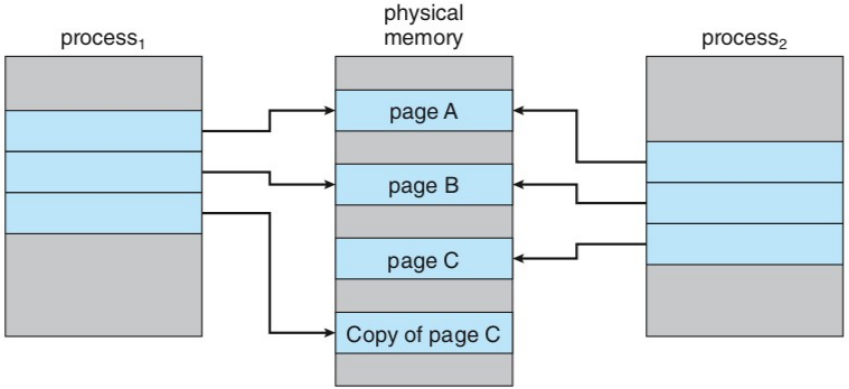

Figure 9.8 After process 1 modifies page C.

References

1. Book -
    i. Modern Operating System – by Tanenbaum
    ii. Operating System Concepts - by Abraham Silberschatz