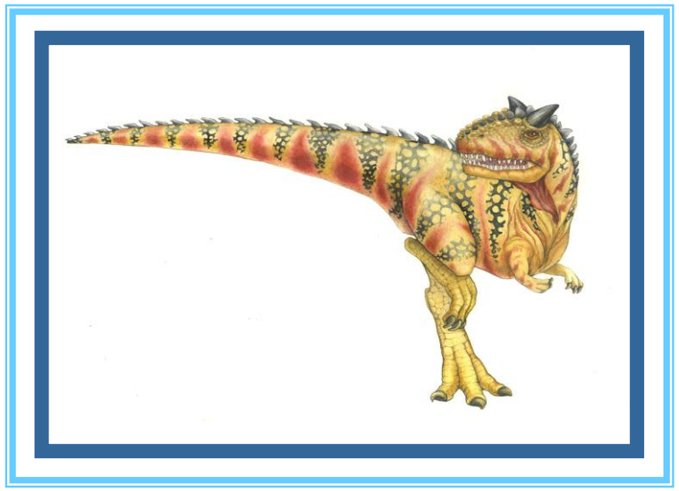
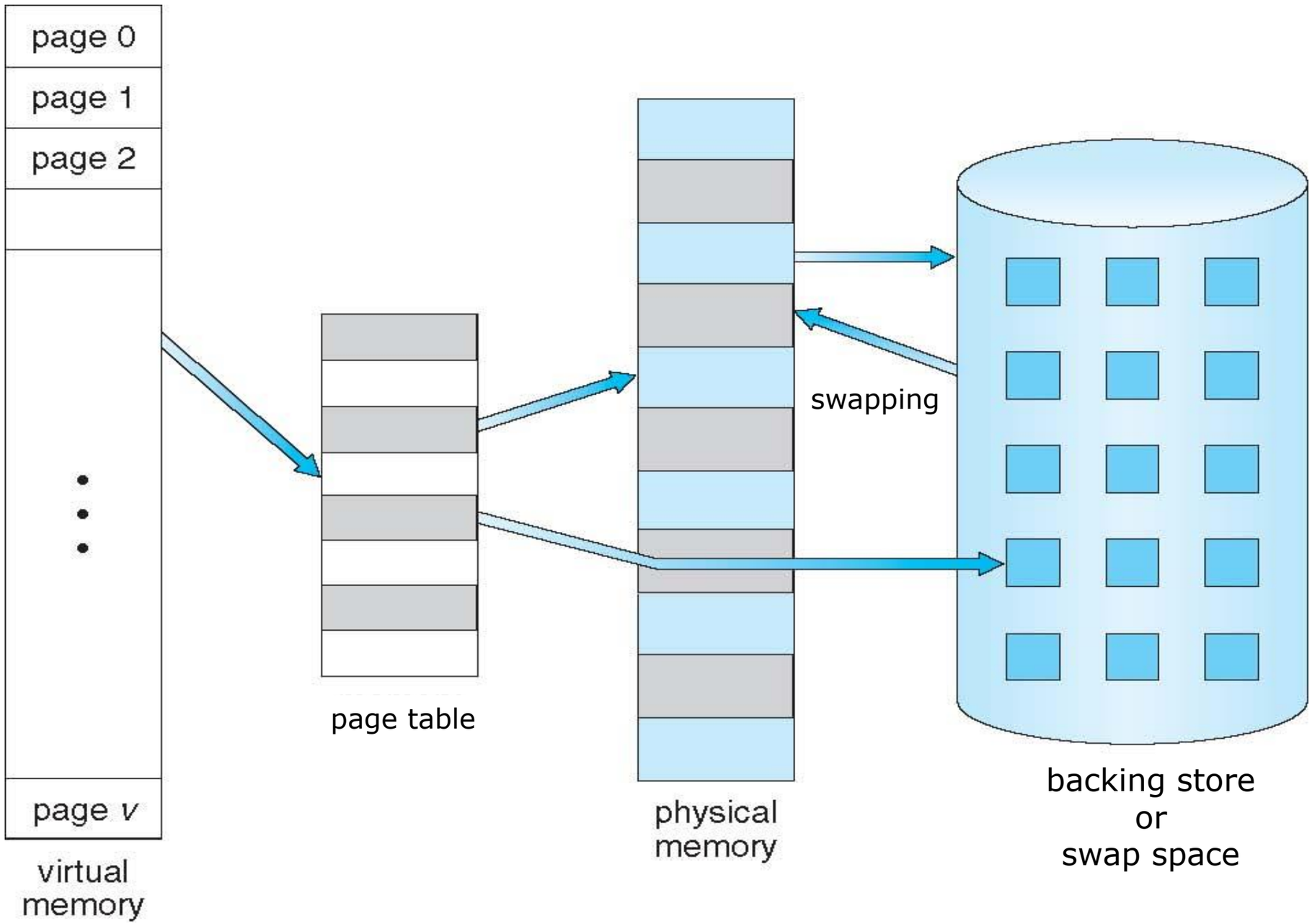
Chapter 9: Virtual-Memory Management



Virtual Memory

- Goal
  - Provide memory space for all processes that can be much larger than physical memory.
- How
  - Only some part of program code and data need to be in physical memory at a time.
    - principle of locality
  - The rest of program code and data can be stored in secondary storage (hard disk)
    - “swap partition” or “swap file”
    - extend memory hierarchy
  - Implement “Demand Paging” mechanism in OS
    - similar to “cache miss”
- Other benefits
  - Address spaces can be shared by several processes
    - shared program code, shared libraries, shared memory (IPC)
  - fast process creation
  - memory-mapped file, memory-mapped I/O

Virtual Memory

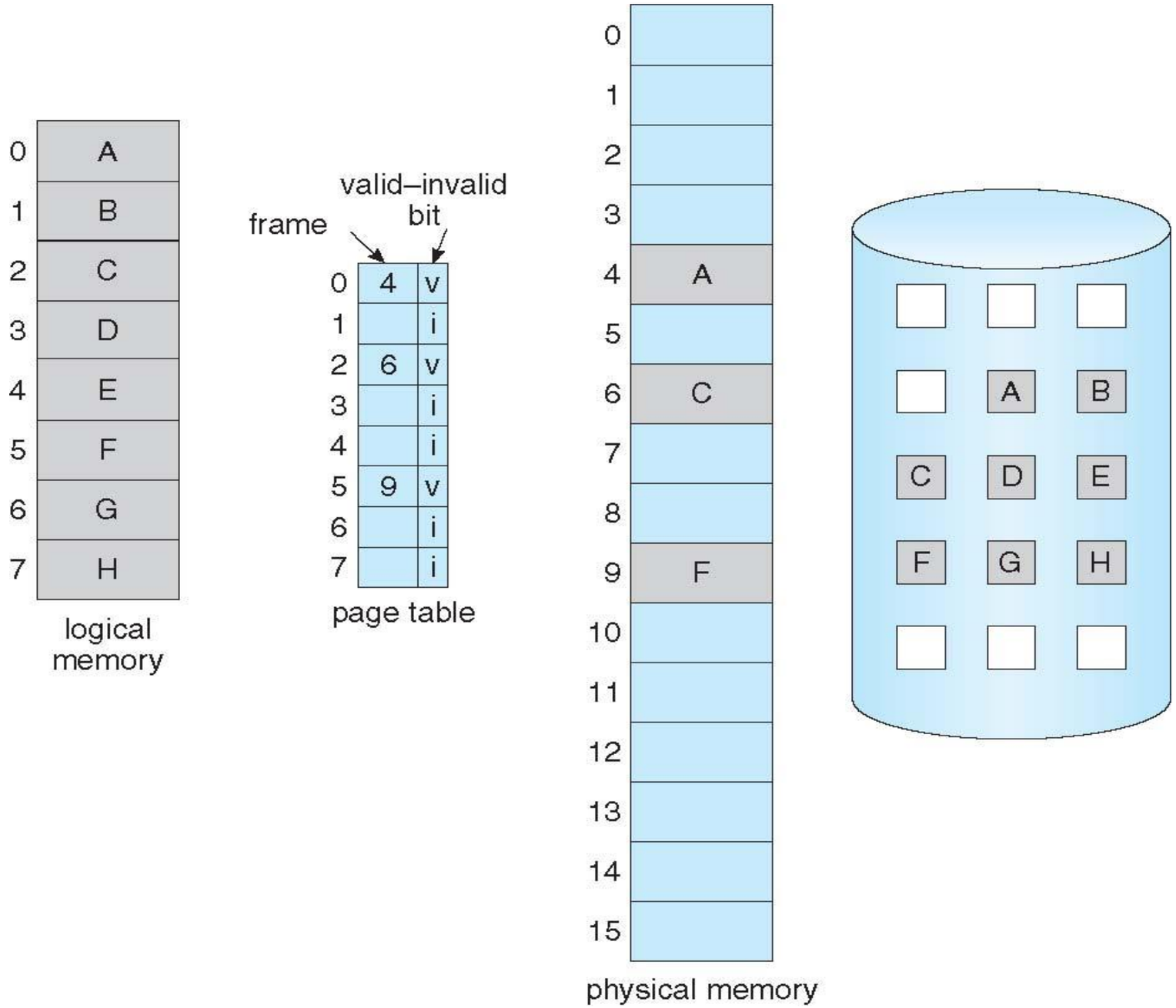


Demand Paging

- Bring a page into memory only when it is needed
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More processes

- CPU instruction references to anywhere in a page ⇒ the page needs to be in physical memory
  - invalid reference ⇒ abort
  - not-in-memory ⇒ bring to memory
  - no-free-frame ⇒ swap

Page table that supports demand paging

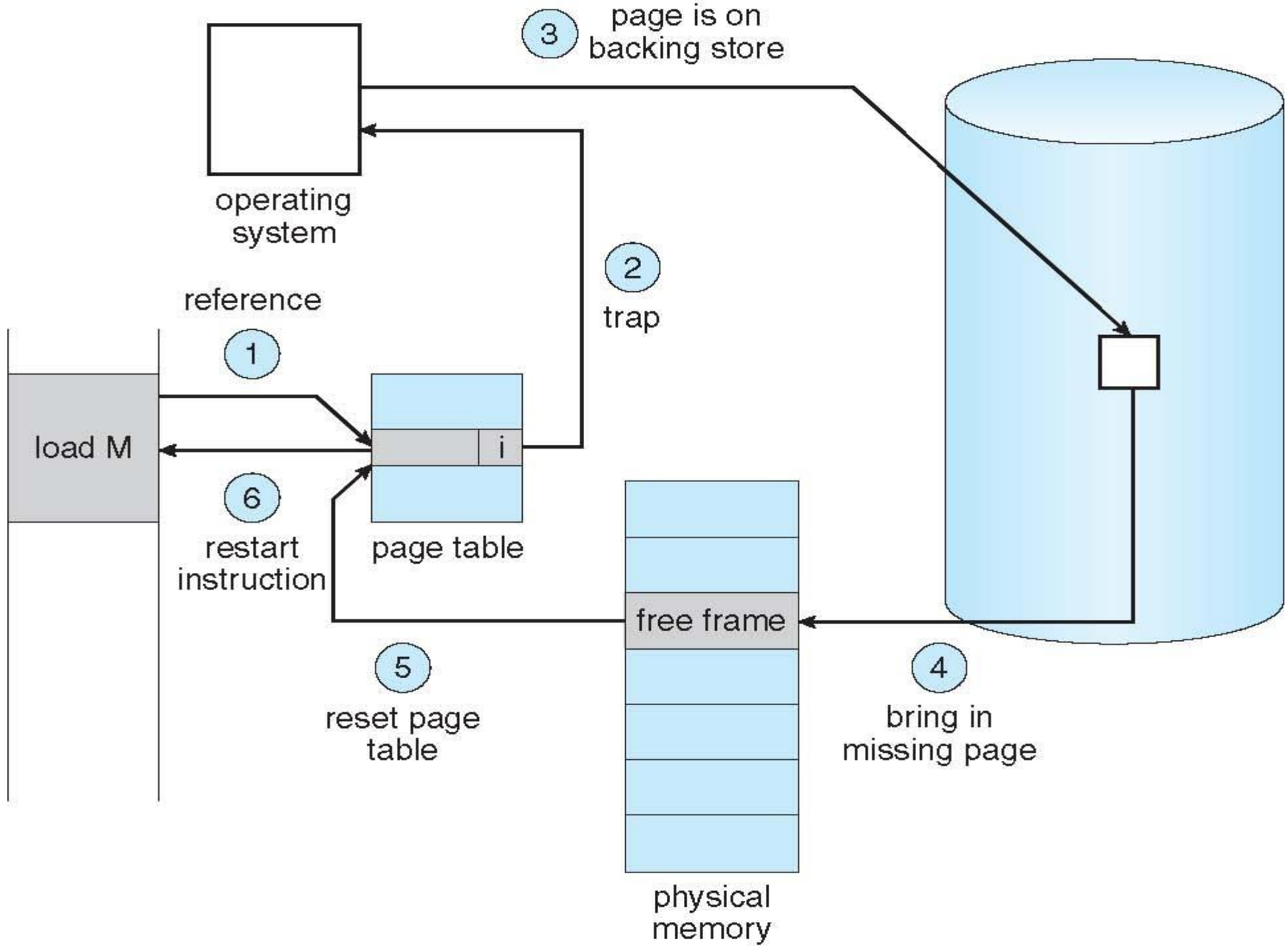


Page Fault

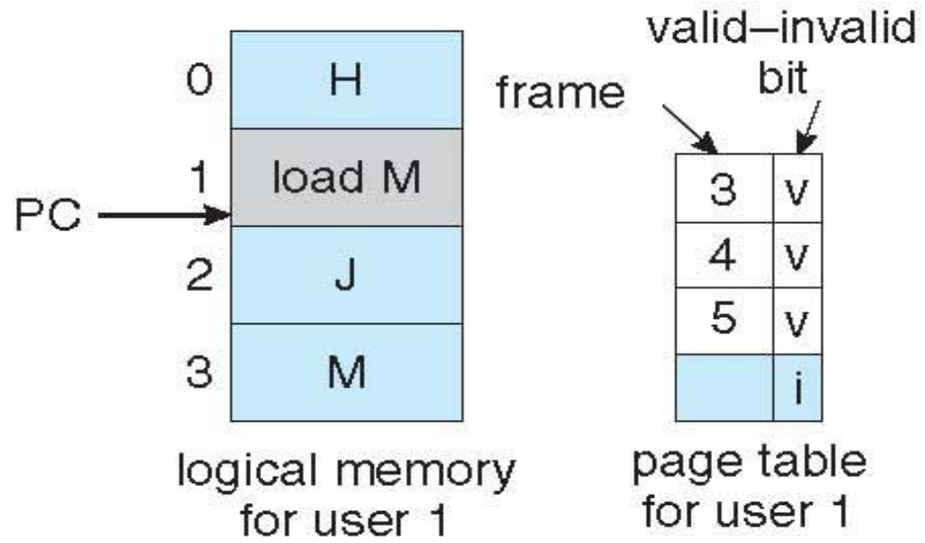
- A reference to an invalid page (e.g. first reference to that page) will cause an interrupt to operating system:
  - page fault interrupt
  - page fault handler

1. Operating system looks at another table to decide:
   - Invalid reference → abort
   - Just not in memory
2. Get empty frame
3. Swap page into frame
4. Update tables
5. Set validation bit = **v**
6. Restart the instruction that caused the page fault

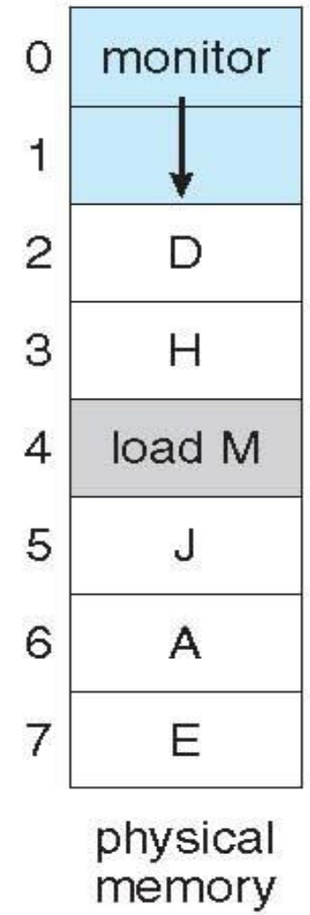
Steps in Handling a Page Fault

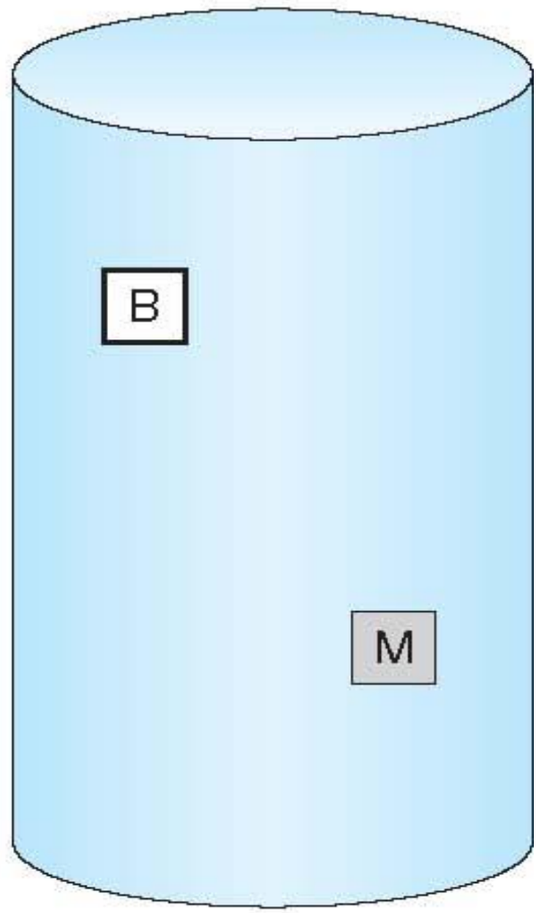


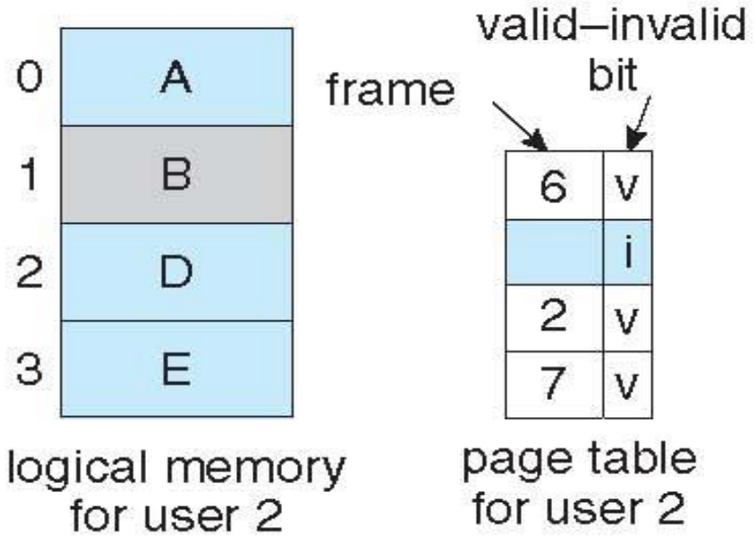
Need For Page Replacement



Program references to M in page 3 which is not in memory.
So, page fault occurs, without a free frame in physical memory.







Basic Page Replacement

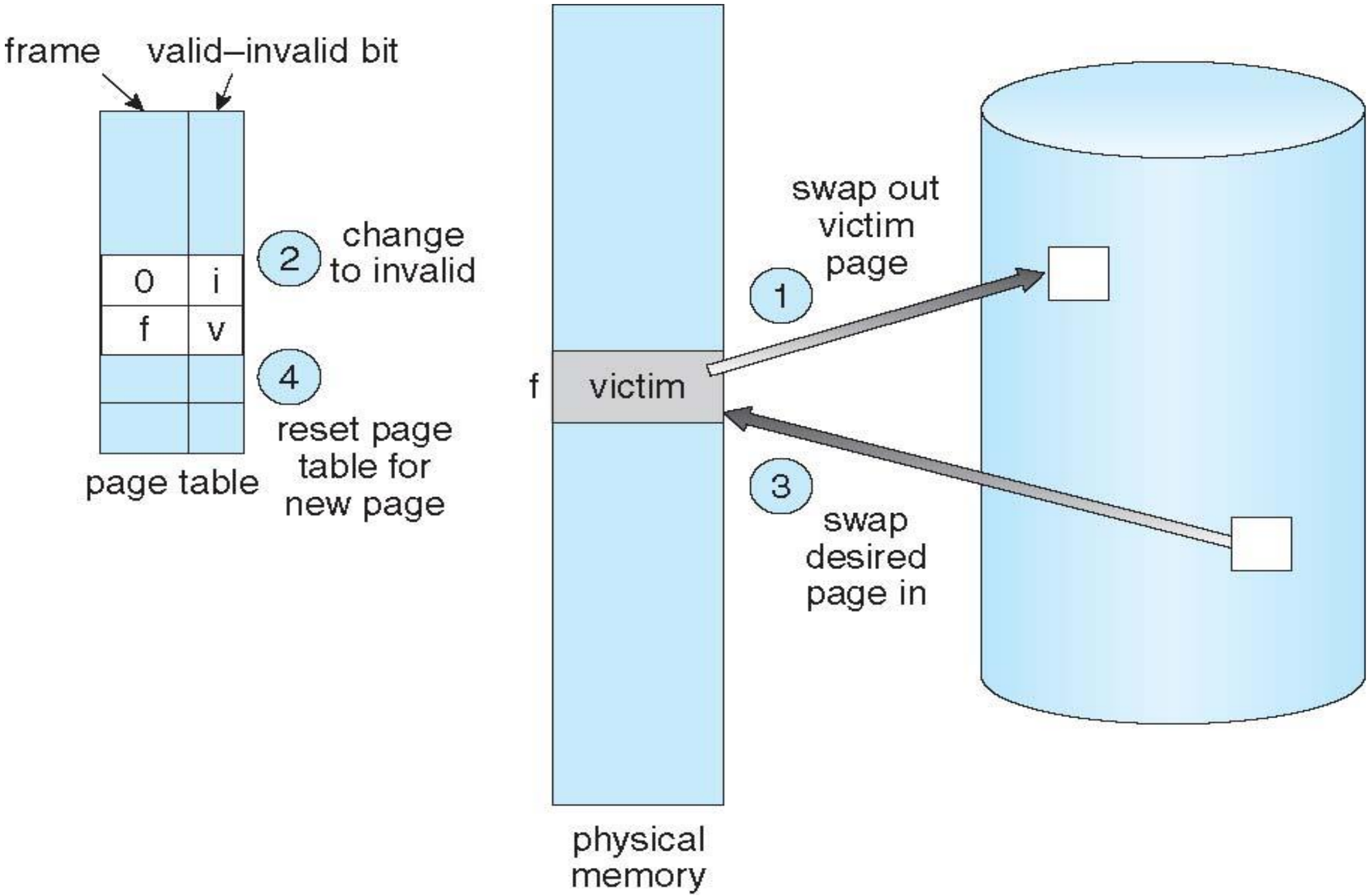
1. Find the location of the desired page on disk

2. Find a free frame:
   - If there is a free frame, use it
   - If there is no free frame, use a page replacement algorithm to select a **victim frame**
     - Write victim frame to disk if dirty

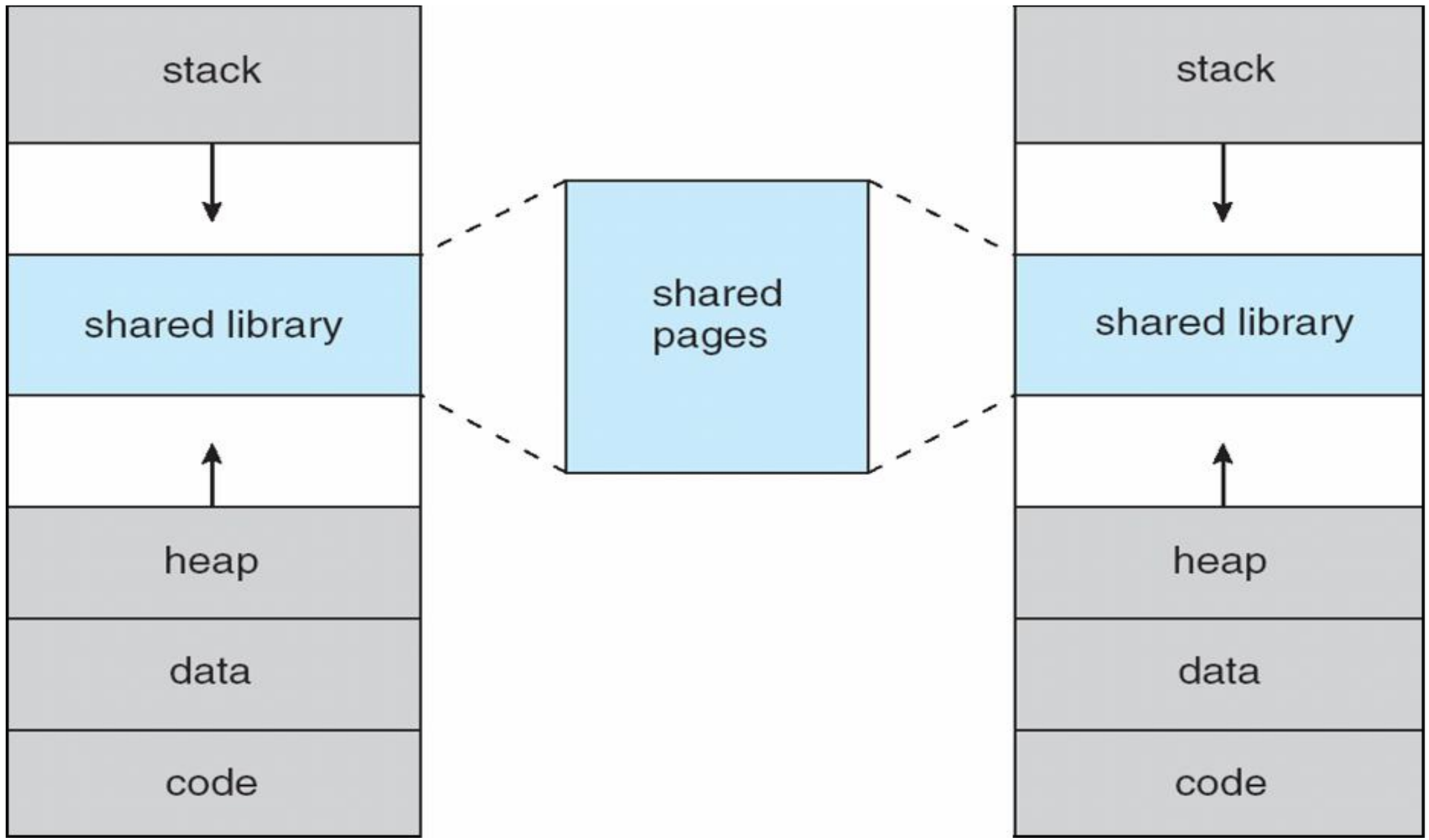
3. Bring the desired page into the (newly) free frame; update the page and frame tables

4. Continue the process by restarting the instruction that caused the trap

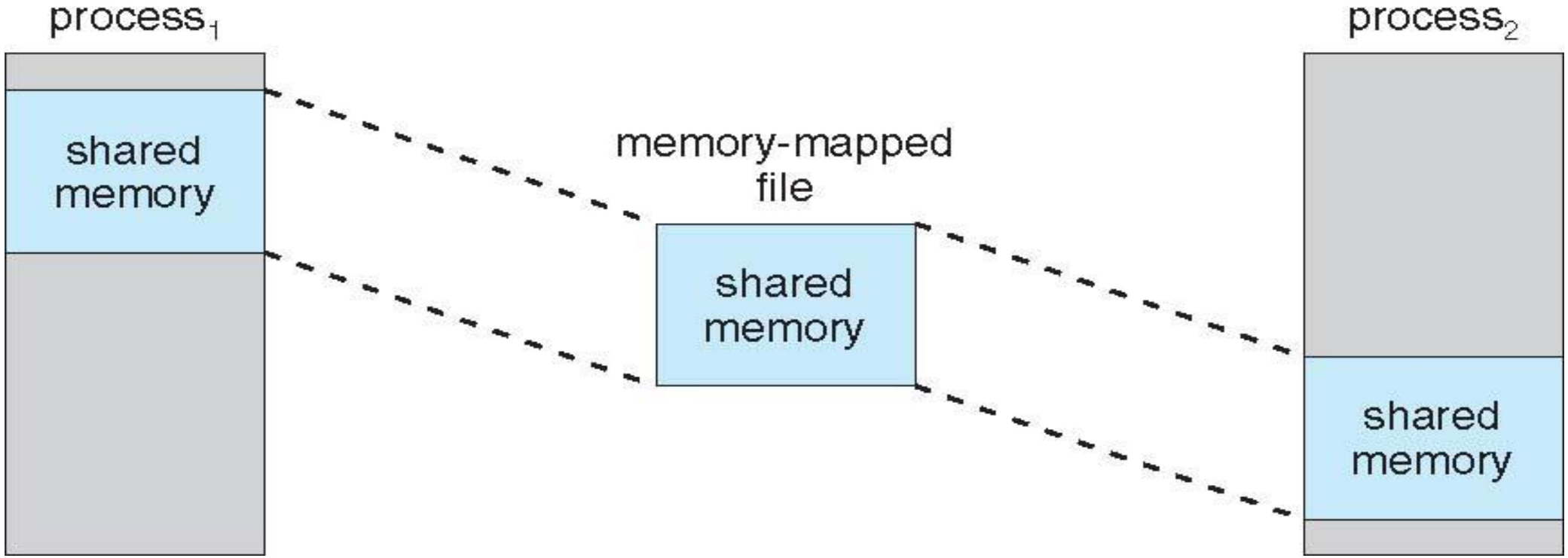
Page Replacement



Shared Library Using Virtual Memory



Memory-Mapped Shared Memory



Memory-Mapped Files

- Memory-mapped file I/O allows file I/O to be treated as routine memory access by **mapping** a disk block to a page in memory
- A file is initially read using demand paging
  - A page-sized portion of the file is read from the file system into a physical page
  - Subsequent reads/writes to/from the file are treated as ordinary memory accesses
- Simplifies and speeds file access by driving file I/O through memory rather than `read()` and `write()` system calls
- Also allows several processes to map the same file allowing the pages in memory to be shared
- But when does written data make it to disk?
  - Periodically and / or at file `close()` time

Memory Mapped Files