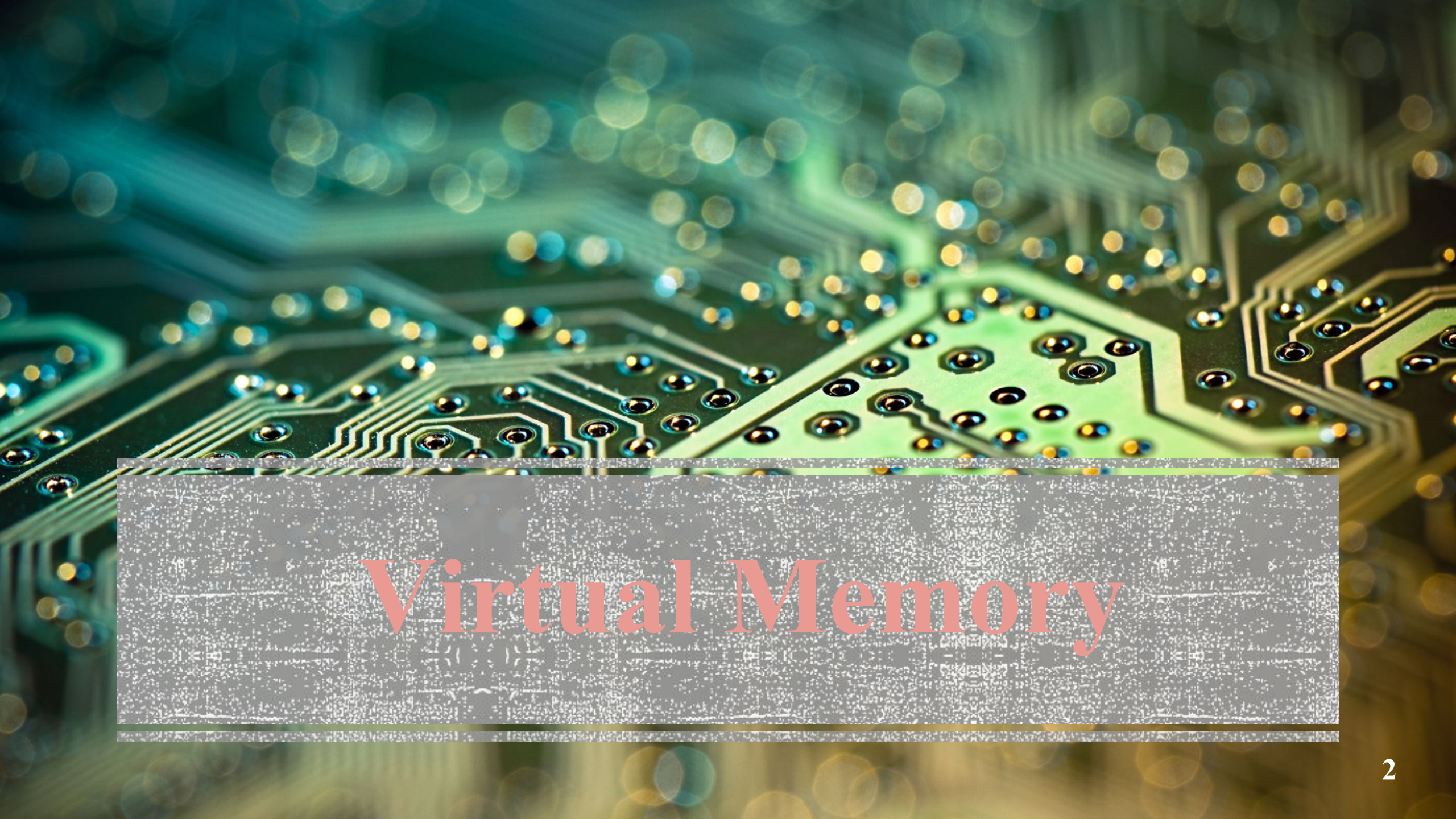
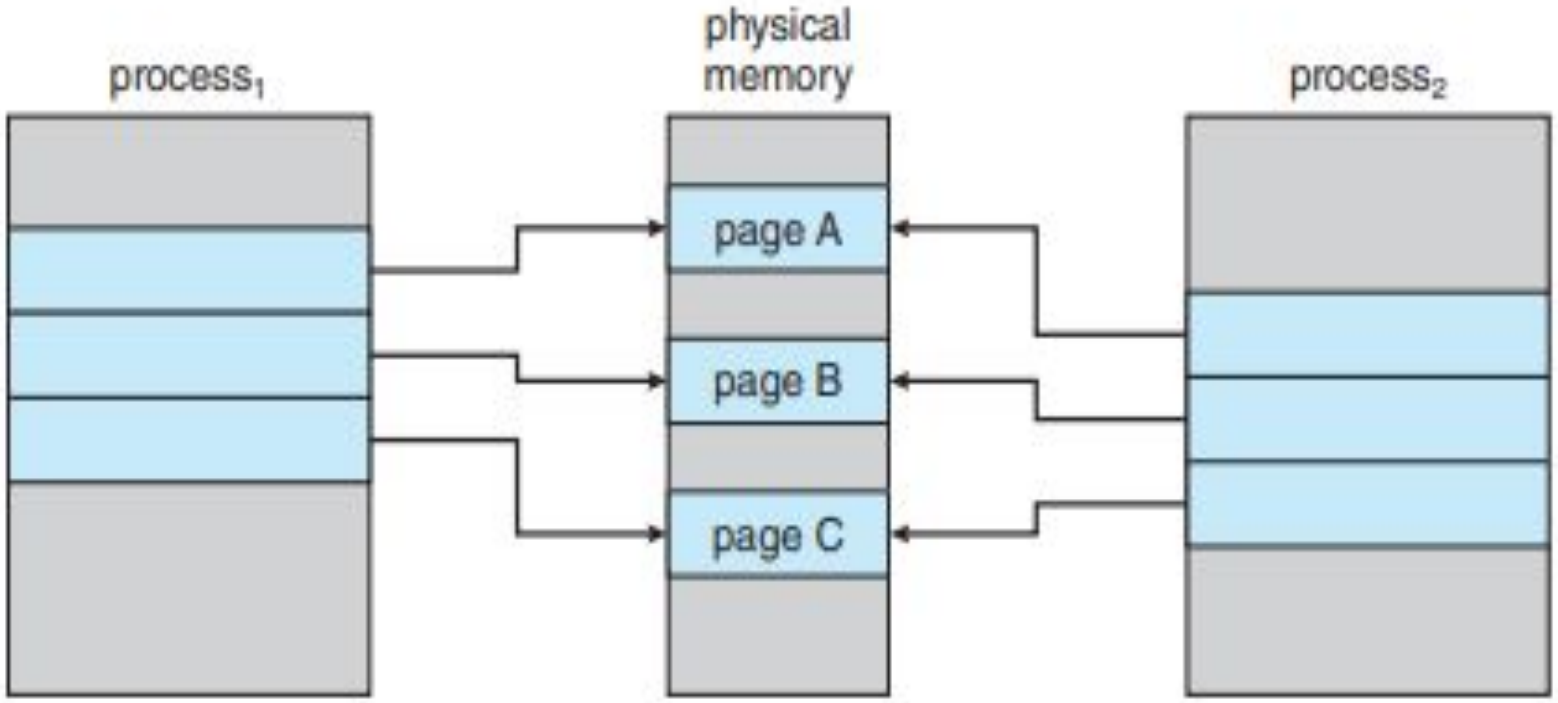
OPERATING SYSTEMS


Virtual Memory

COPY-ON-WRITE

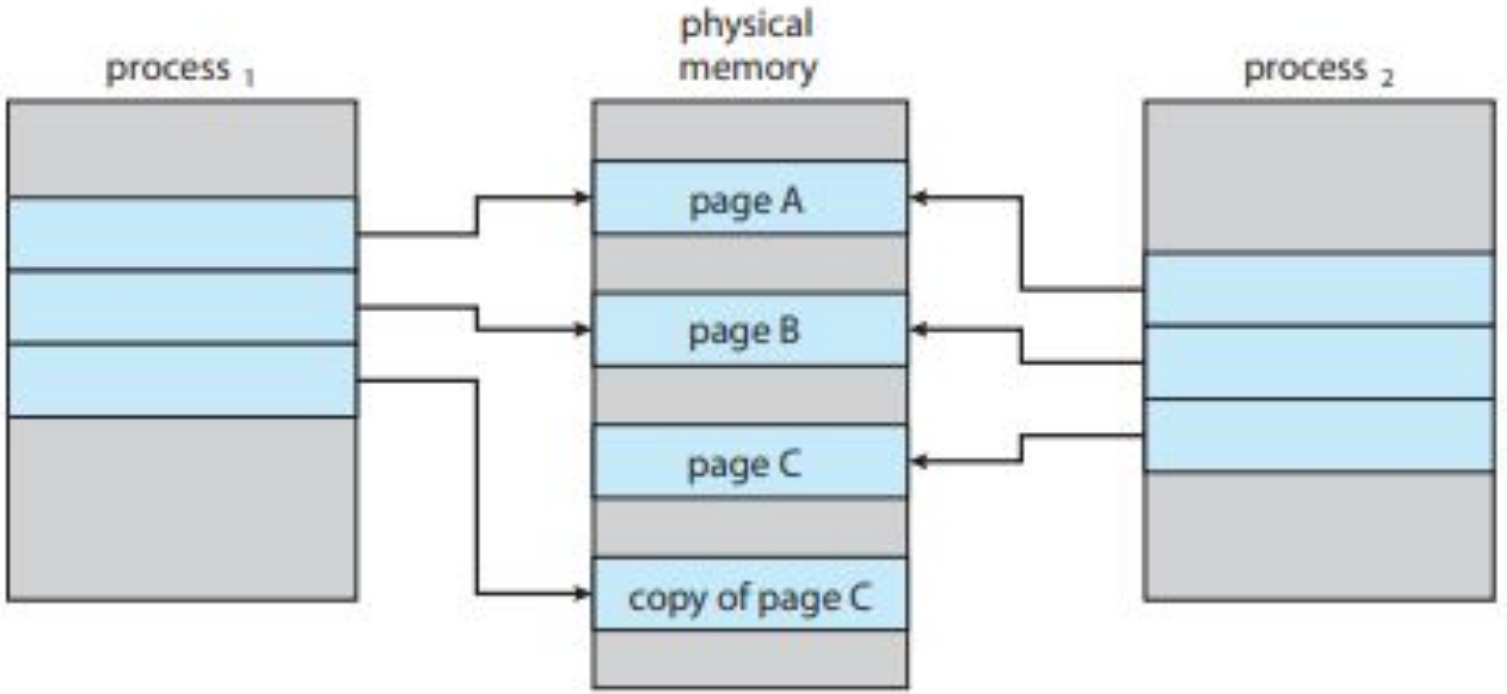
- fork() system call creates a child process that is a duplicate of its parent.
- Traditionally, fork() worked by creating a copy of the parent's address space for the child, duplicating the pages belonging to the parent.
- However, considering that many child processes invoke the exec() system call immediately after creation, the copying of the parent's address space may be unnecessary.
- Instead, we can use a technique known as copy-on-write, which works by allowing the parent and child processes initially to share the same pages.
- These shared pages are marked as copy-on-write pages, meaning that if either process writes to a shared page, a copy of the shared page is created.

COPY-ON-WRITE



Before process 1 modifies page C.

COPY-ON-WRITE



After process 1 modifies page C.

COPY-ON-WRITE

- For example, assume that the child process attempts to modify a page containing portions of the stack, with the pages set to be copy-on-write.
- The operating system will obtain a frame from the free-frame list and create a copy of this page, mapping it to the address space of the child process.
- The child process will then modify its copied page and not the page belonging to the parent process.
- Obviously, when the copy-on-write technique is used, only the pages that are modified by either process are copied; all unmodified pages can be shared by the parent and child processes.
- Note, too, that only pages that can be modified need be marked as copy-on-write.
- Pages that cannot be modified (pages containing executable code) can be shared by the parent and child.
- Copy-on-write is a common technique used by several operating systems, including Windows, Linux, and macOS.

VFORK()

- Several versions of UNIX (including Linux, macOS, and BSD UNIX) provide a variation of the fork() system call—vfork() (for virtual memory fork)— that operates differently from fork() with copy-on-write
- With vfork(), the parent process is suspended, and the child process uses the address space of the parent.
- Because vfork() does not use copy-on-write, if the child process changes any pages of the parent's address space, the altered pages will be visible to the parent once it resumes.
- Therefore, vfork() must be used with caution to ensure that the child process does not modify the address space of the parent.
- vfork() is intended to be used when the child process calls exec() immediately after creation.
- Because no copying of pages takes place, vfork() is an extremely efficient method of process creation and is sometimes used to implement UNIX command-line shell interfaces.

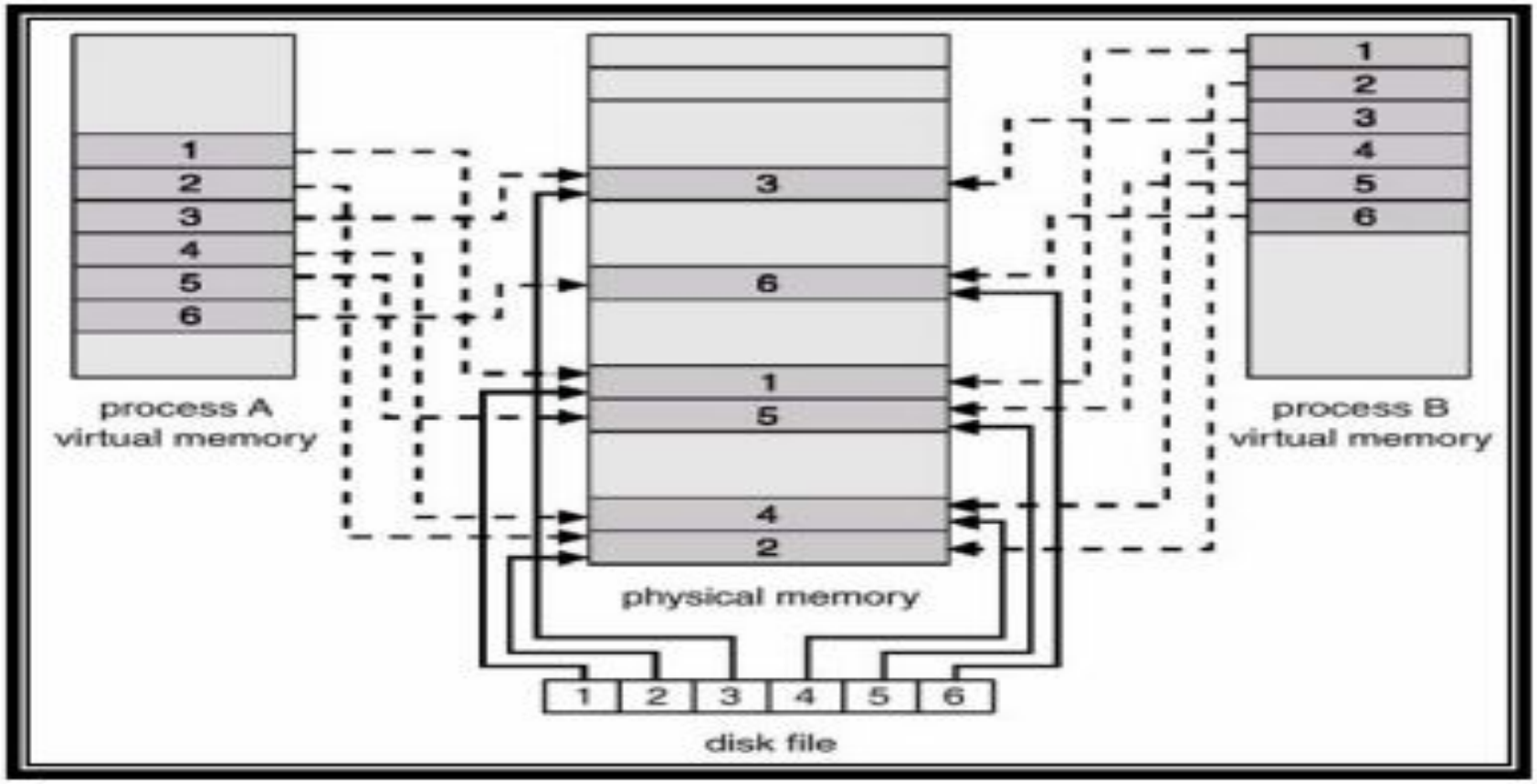
MEMORY MAPPED FILES

- Consider a sequential read of a file on disk using the standard system calls open(), read(), write().
- Every time the file is accessed requires a system call and disk access.
- Alternatively, we can use the virtual memory techniques discussed so far to treat file I/O as routine memory accesses.
- This approach is known as memory mapping a file, allowing a part of the virtual address space to be logically associated with a file.
- Memory mapping a file is possible by mapping a disk block to a page (or pages) in memory.
- Initial access to the file proceeds using ordinary demand paging resulting in a page fault.
- However, a page sized portion of the file is read from the file system into a physical page.

MEMORY MAPPED FILES

- Subsequent reads and writes to the file are handled as routine memory accesses, thereby simplifying file access and usage by allowing file manipulation through memory rather than the overhead of using the read() and write() system calls.
- Note that writes to the file mapped in memory may not be immediate writes to the file on disk.
- Some systems may choose to update the physical file when the operating system periodically checks if the page in memory mapping the file has been modified.
- Closing the file results in all the memory mapped data being written back to disk and removed from the virtual memory of the process.

MEMORY MAPPED FILES



Memory mapped files

MEMORY MAPPED FILES

- Multiple processes may be allowed to map the same file into the virtual memory of each to allow sharing of data.
- Writes by any of the processes modify the data in virtual memory and can be seen by all others that map the same section of the file.
- The memory mapping system calls can only support copy-on-write functionality allowing processes to share a file in read-only mode, but to have their own copies of any data they modify.
- So that access to the shared data is coordinated, the processes involved might use one of the mechanisms for achieving mutual exclusion.