CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Memory-Mapped Files

Virtual memory has yet another benefit not mentioned thus far: it can make file I/O significantly more convenient and efficient

In standard I/O, each `read` or `write` requires access to disk

Disk access has poor latency, but somewhat better throughput

Accessing disk many times for small things is worse than accessing once for something larger

Memory mapping is used to treat I/O as though it were memory access

File treated as part of logical address space

Will see that this can be more convenient and much more efficient

OS associates block of file (on disk) with page of memory

First time block is accessed, page fault occurs and page worth (or more) of file read *into memory*

Next time file is read, if new read address was also pulled in with page, going to disk is not necessary

Subequent reads from this block can occur directly from memory

Much faster than going out to disk each time with `read` and `write`

Also, may be easier to access file through memory access (such as pointers) than through `read` and `write` `writes` to file generally not propogated directly to disk

Write to memory for efficiency, then flush changes to disk when file is closed

Much more efficient because

- may write to same location more than once -- no need for disk to see overwritten changes
- writing all changes at once faster than many writes (throughput vs latency)

User can request file be memory mapped with system call such as `mmap`

In that case, file will be mapped to process memory and can be accessed as pointer

Some OSes memory map *every* file access, regardless of whether user requests it

In this case, file mapped to OS memory and user is unaware that disk not directly accessed

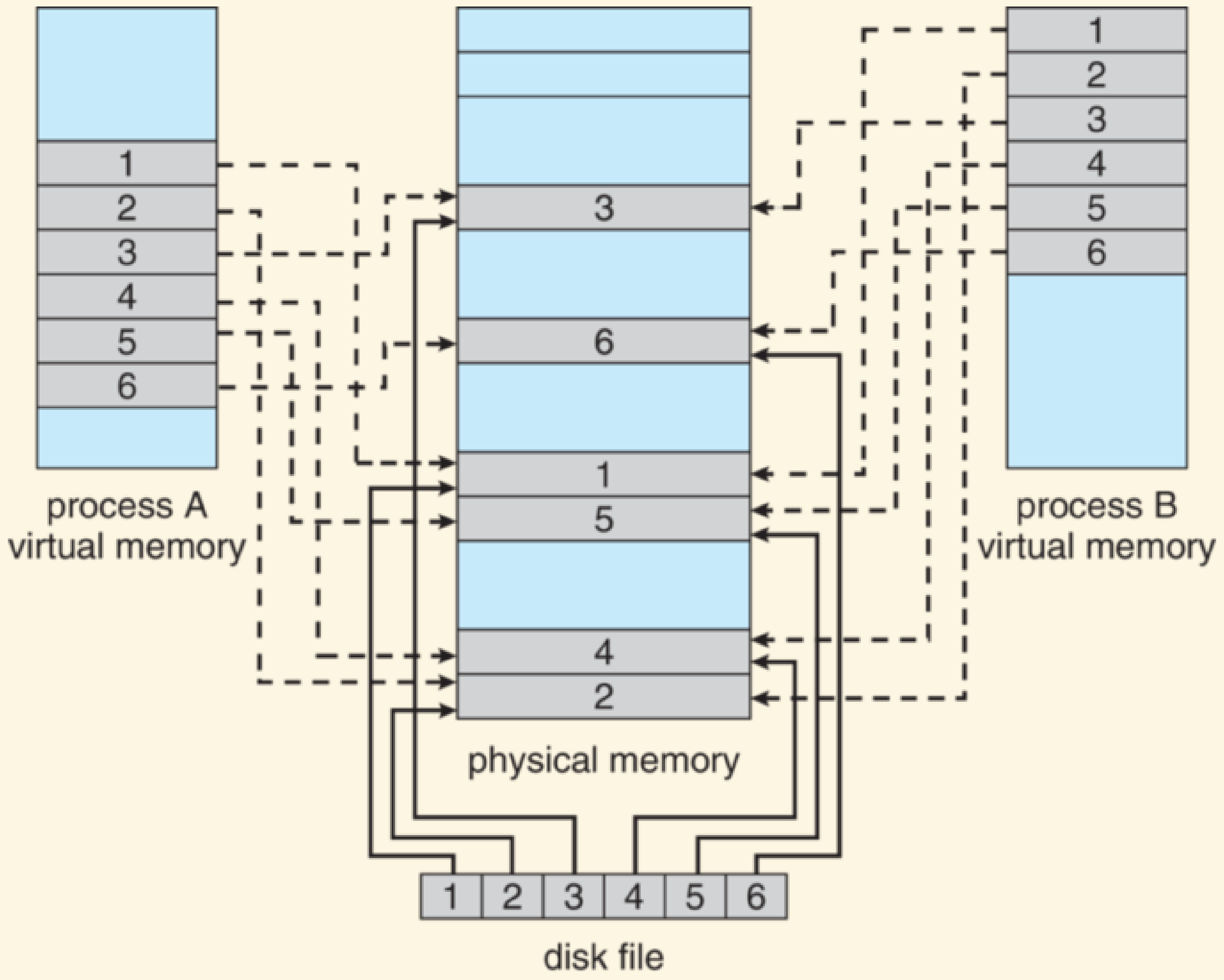
User would still access with `read`/`write` as always, but OS would manipulate file in memory

This is done to take advantage of improved efficiency of memory-mapped files

Because of virtual memory, sharing memory-mapped file is simple

We have already seen how to share memory in virtual-memory system; sharing memory-mapped file is no different

Simply requires updating page tables of processes involved


1
2
3
4
5
6
process A virtual memory
3
6
1
5
4
2
physical memory
1
2
3
4
5
6
process B virtual memory
1 2 3 4 5 6
disk file

In fact, you have seen shared files used to implement shared memory

POSIX shared memory example from earlier in course used shared memory-mapped file as shared memory for processes

Windows can use shared files to implement shared memory for processes in similar way