CISC 3320
C18b Swapping

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Acknowledgement

- These slides are a revision of the slides provided by the authors of the textbook via the publisher of the textbook

Outline

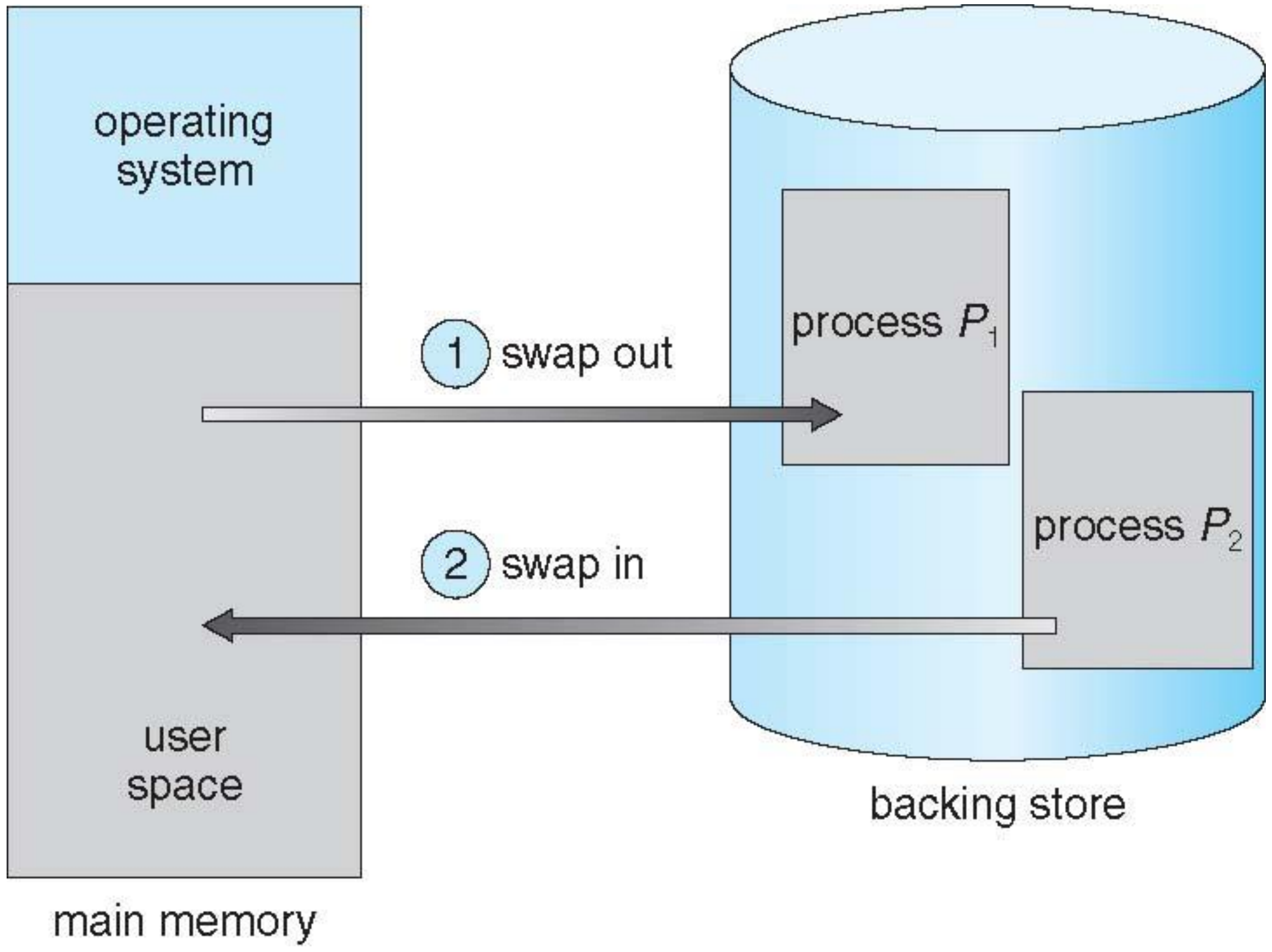
- Swapping

- Example: The Intel 32 and 64-bit Architectures
- Example: ARMv8 Architecture

Swapping

- Total physical memory space of processes can exceed physical memory
- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution
  - Backing store – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
  - Roll out, roll in – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed

Schematic View of Swapping



Overhead (Cost) of Swapping

- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped

Ready Queue

- System maintains a ready queue of ready-to-run processes which have memory images on disk

Design Issues of Swapping

- Does the swapped out process need to swap back in to same physical addresses?
- Depends on address binding method
  - Plus consider pending I/O to / from process memory space

Context Switch Time including Swapping

- If next processes to be put on CPU is not in memory, need to swap out a process and swap in target process
- Context switch time can then be very high
- 100MB process swapping to hard disk with transfer rate of 50MB/sec
  - Swap out time of 2000 ms
  - Plus swap in of same sized process
  - Total context switch swapping component time of 4000ms (4 seconds)
- Can reduce if reduce size of memory swapped – by knowing how much memory really being used
  - System calls to inform OS of memory use via `request_memory()` and `release_memory()`

Context Switch Time and Swapping

- Other constraints as well on swapping
  - Pending I/O – can't swap out as I/O would occur to wrong process
  - Or always transfer I/O to kernel space, then to I/O device
    - Known as **double buffering**, adds overhead
- Standard swapping not used in modern operating systems
  - But modified version common
    - Swap only when free memory extremely low

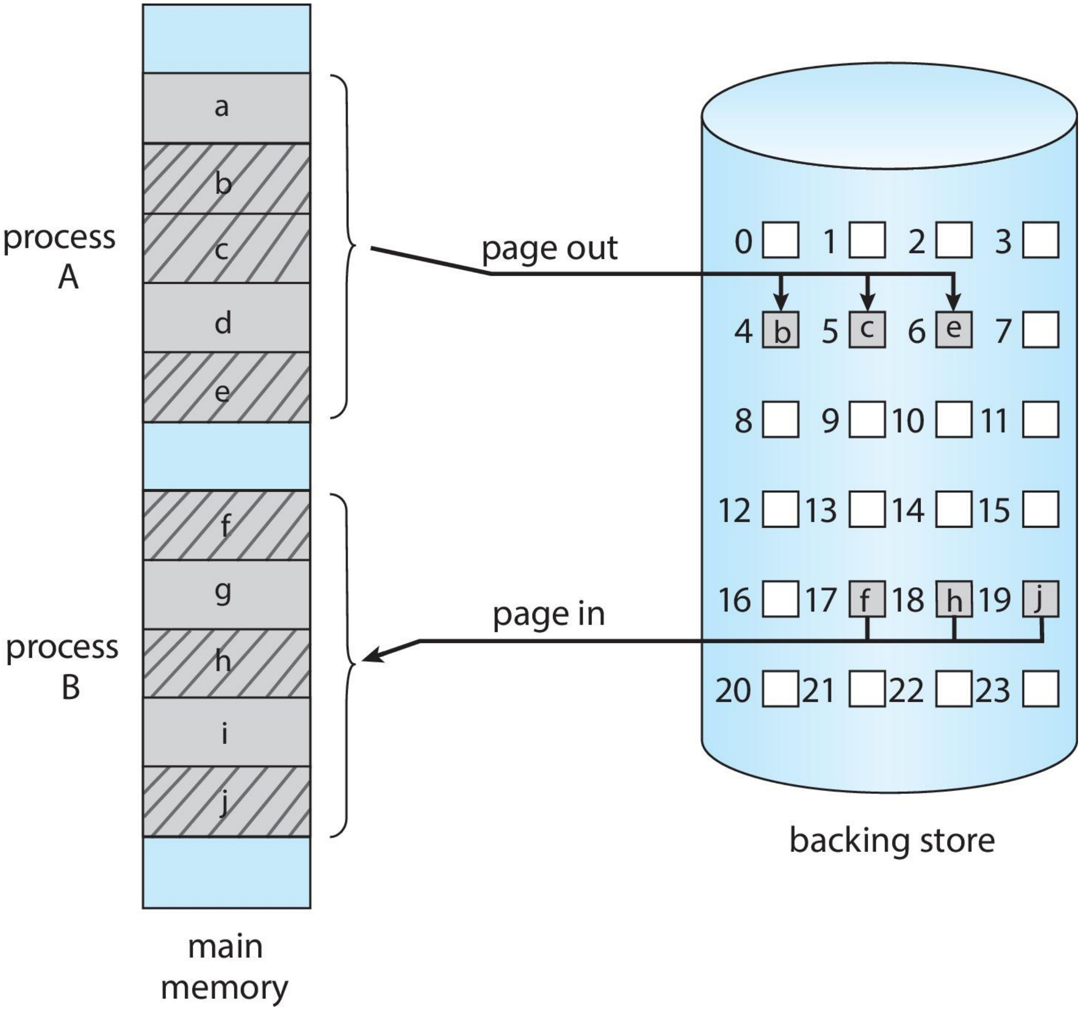
Modified Swapping

- Modified versions of swapping are found on many systems (i.e., UNIX, Linux, and Windows)
  - Swapping normally disabled
  - Started if more than threshold amount of memory allocated
  - Disabled again once memory demand reduced below threshold

Swapping on Mobile Systems

- Not typically supported
  - Flash memory based
    - Small amount of space
    - Limited number of write cycles
    - Poor throughput between flash memory and CPU on mobile platform
- Instead use other methods to free memory if low
  - iOS ***asks*** apps to voluntarily relinquish allocated memory
    - Read-only data thrown out and reloaded from flash if needed
    - Failure to free can result in termination
  - Android terminates apps if low free memory, but first writes **application state** to flash for fast restart
  - Both OSes support paging as discussed below

Paging (Swapping with Paging)



Questions?

- Swapping?
    - In fact, the term swapping now generally refers to standard swapping, and paging refers to swapping with paging