Department of Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY
UNIVERSITY OF LUCKNOW
LUCKNOW

CS-501

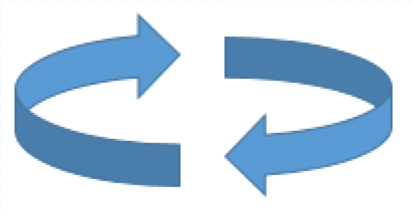
Dr. Zeeshan Ali Siddiqui

Assistant Professor

Deptt. of C.S.E.

SWAPPING
AND
MEMORY ALLOCATION

SWAPPING



Swapping

- A *process* can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.

- **Backing store**
  - 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.

Swapping

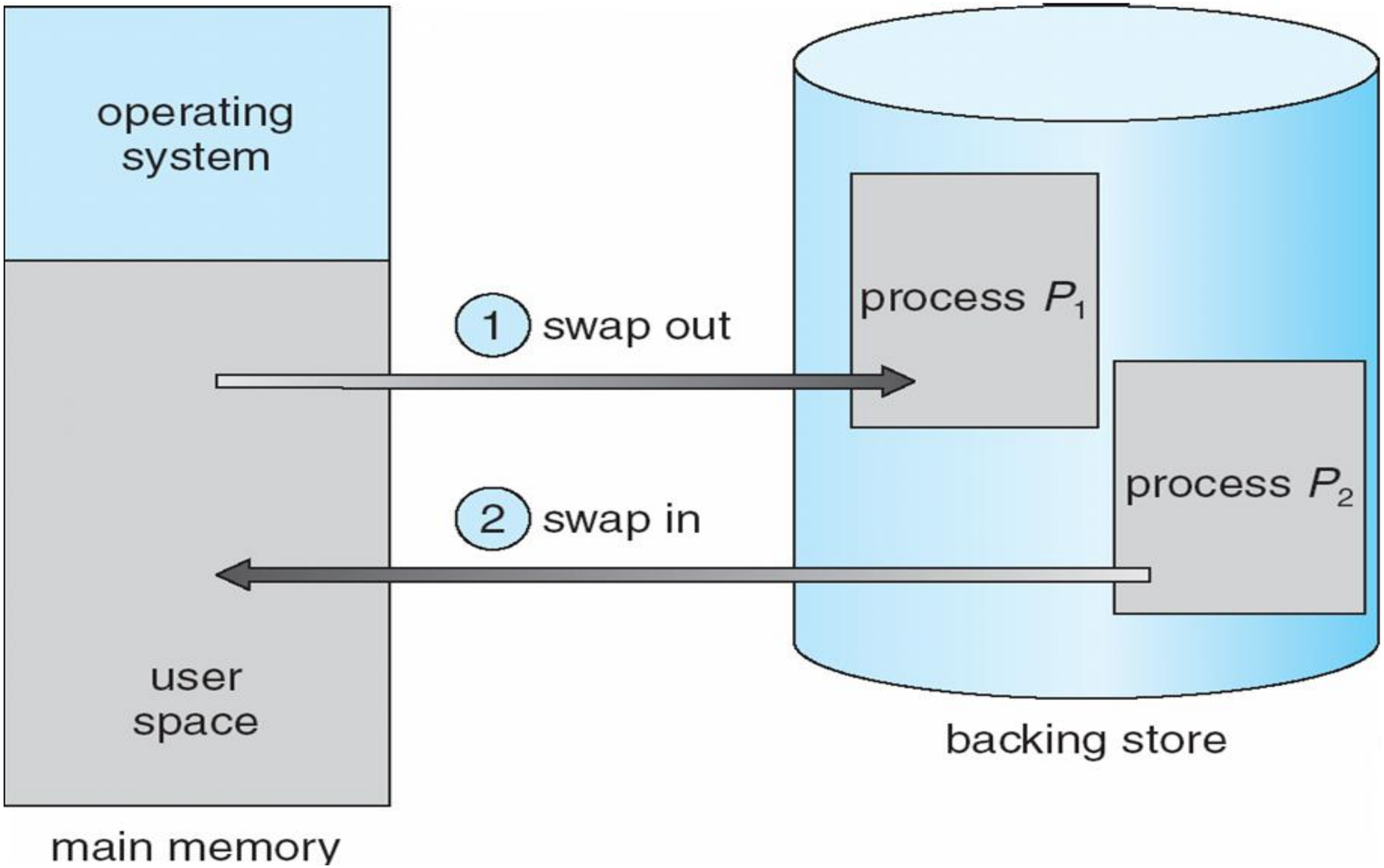
- Major part of swap time is *transfer time*.

- Total transfer time is directly proportional to the amount of *memory* swapped.

- Modified versions of *swapping* are found on many systems (i.e., UNIX, Linux, and Windows).

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

Swapping


operating system
user space
main memory
1 swap out
2 swap in
process P_1
process P_2
backing store

MEMORY ALLOCATION

Memory Allocation

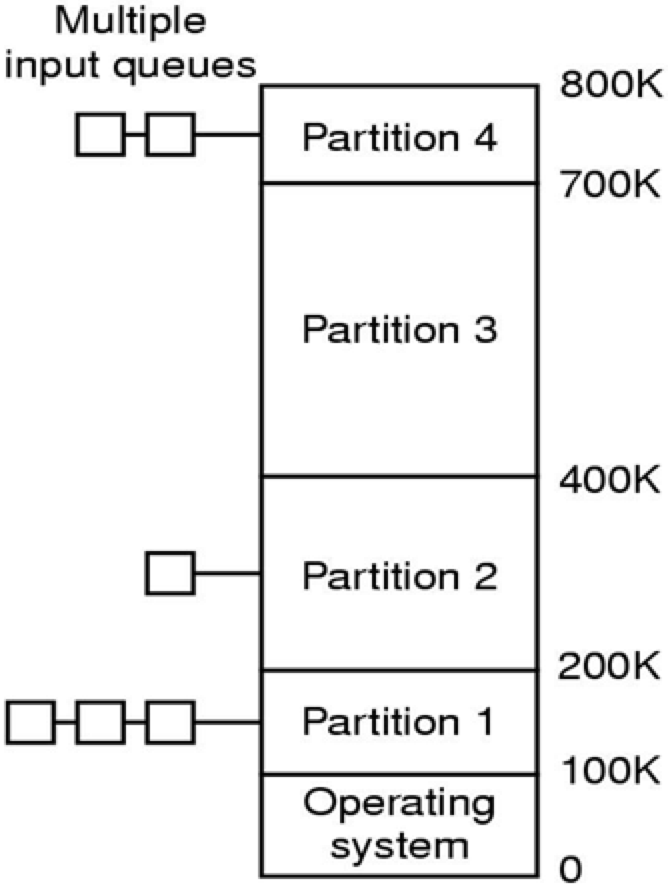
- **Problem**:
  - How to allocate memory for multiple *processes* (in a multi-programming environment).

- **Solutions:**
- Contiguous allocation
  - Fixed partitions
  - Dynamic partitions

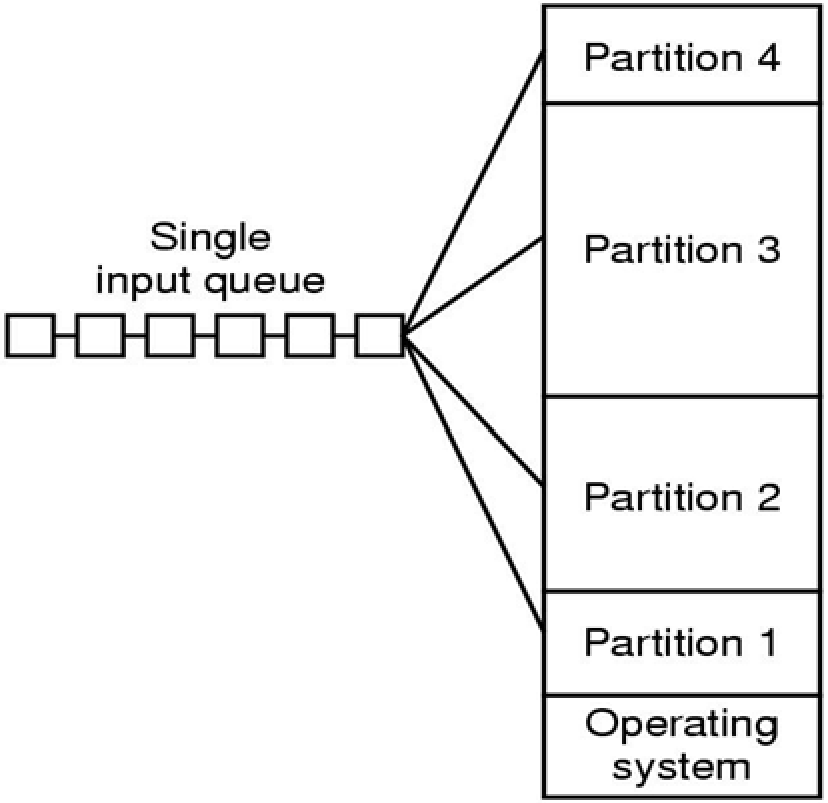
- Paging

Contiguous Allocation (Fixed Partitions)

Contiguous Allocation (Fixed Partitions)



(a)



(b)

Contiguous Allocation (Dynamic Partitions)

Contiguous Allocation (Dynamic Partitions):

- *Main memory* usually have two partitions:
  - Resident operating system, usually held in *low memory* with interrupt vector.
  - User processes, held in *high memory*.

Contiguous Allocation (Dynamic Partitions):

- Multiple-partition allocation
  - **Hole**
    - Block of available *memory*.
    - Holes of various size are *scattered* throughout memory.
  - When a *process* arrives, it is allocated memory from a large enough hole.
  - Operating system maintains information about:
    a) Allocated partitions
    b) Free partitions (hole).

Dynamic Storage-Allocation Problems

Dynamic Storage-Allocation Problems

- **Problem**
  - How to satisfy a request of size n from a list of free holes.

Dynamic Storage-Allocation Problems

- **Solution**:
  - *First-fit:* Allocate the first hole that is big enough.
  - *Best-fit:* Allocate the smallest hole that is big enough; must search entire list, unless ordered by size
    - Produces the smallest leftover hole
  - *Worst-fit:* Allocate the largest hole; must also search entire list
    - Produces the largest leftover hole
- **Note**: First-fit and best-fit better than worst-fit in terms of *speed and storage utilization*.

Question-1

- Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Rank the algorithms in terms of how efficiently they use memory.

Question-2

- Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)? Rank the algorithms in terms of how efficiently they use memory.

References

1. Silberschatz, Galvin and Gagne, “Operating Systems Concepts”, Wiley.

2. William Stallings, “Operating Systems: Internals and Design Principles”, 6th Edition, Pearson Education.

3. D M Dhamdhere, “Operating Systems: A Concept based Approach”, 2nd Edition, TMH.

Thank You.