5.4 Explores how an operating system manages the resources

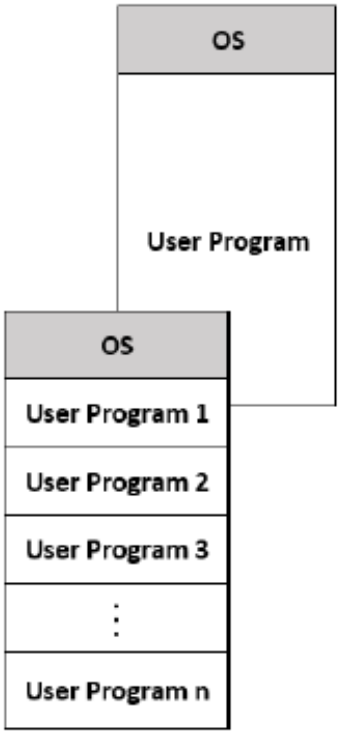
Time: 6 periods

Learning Outcomes

- Briefly explains the need of memory management and memory management unit (MMU).
- Briefly explains the virtual memory.
- Briefly explains paging and mapping
- Briefly describes how an OS manages Input and output devices
- Briefly describes device drivers
- Briefly describes the need of device drivers
- Briefly describes spooling
- Installs appropriate device drivers when connecting a peripheral

Memory Management

Sub division (**partitioning**) of memory is called memory management. In **uniprogramming**, memory is divided into a partition for OS and a portion for user program being executed. In **multiprogramming** user program is sub divided to accommodate multiple programs being executed concurrently.

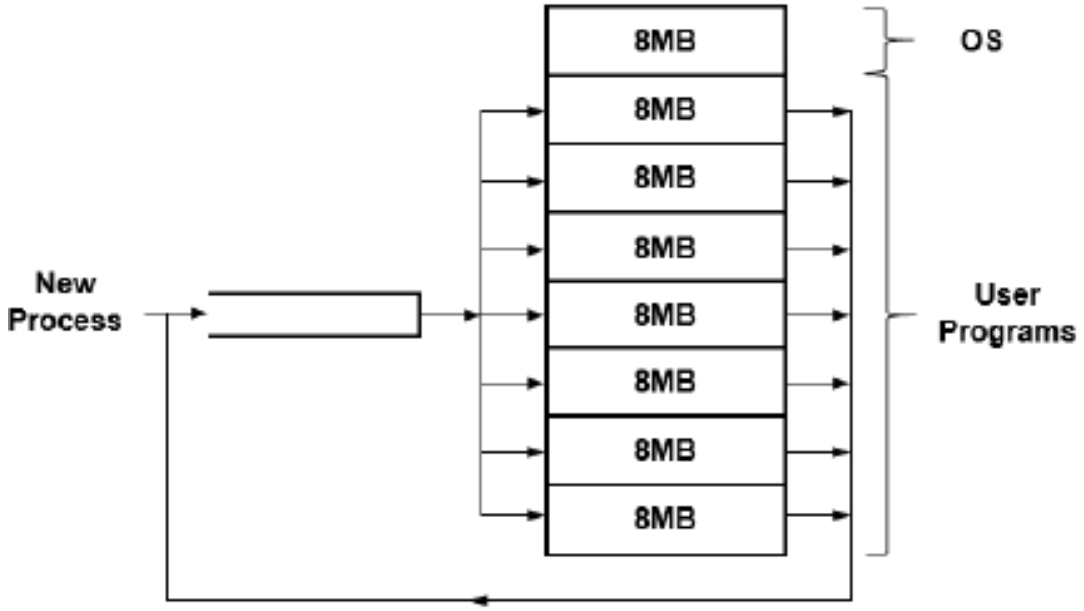


Fixed Partitioning

- Partitions memory into **fixed static** portions during the design of the system
- Fixed number of processes can use memory at a time
- Two kinds of fixed partitioning
    - Equal size fixed partitioning
    - Unequal size fixed partitioning

Equal size fixed partitioning

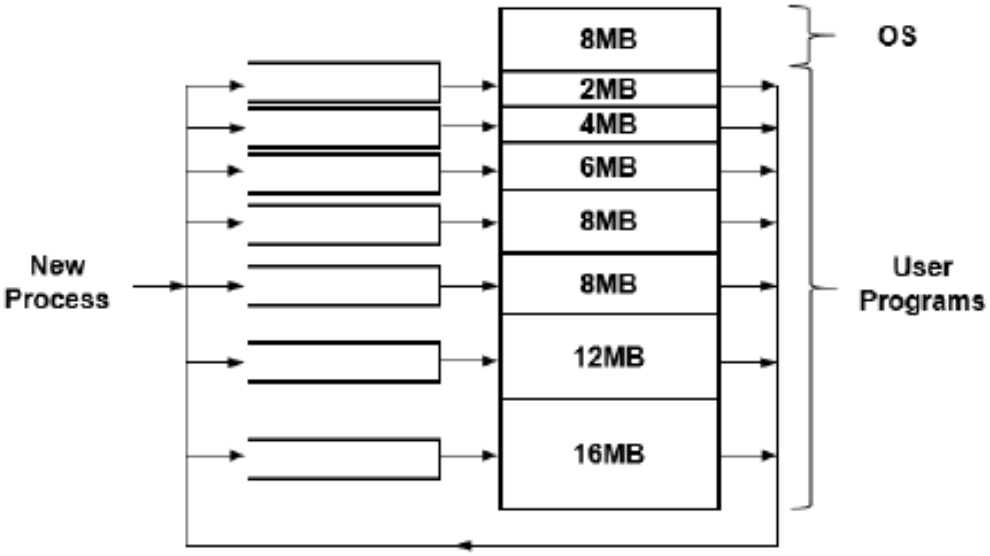
- Incoming process is directed to any available partition that is big enough to load it.
- If all partitions are occupied and all processes are not in ready state, one of them is swapped out and a new process is swapped in.



- Strengths
    - Simple to implement
    - Minimum overheads
- Weaknesses
    - If process is too small, it may not utilize the partition space efficiently (**internal fragmentation**).
    - If process is too large for the partition, program is required to redesign.

Unequal size fixed partitioning

- Incoming process is directed to the smallest available partition that is big enough to load it
- Every partition has a separate queue to hold the swapped out processes suitable for that partition



Strengths

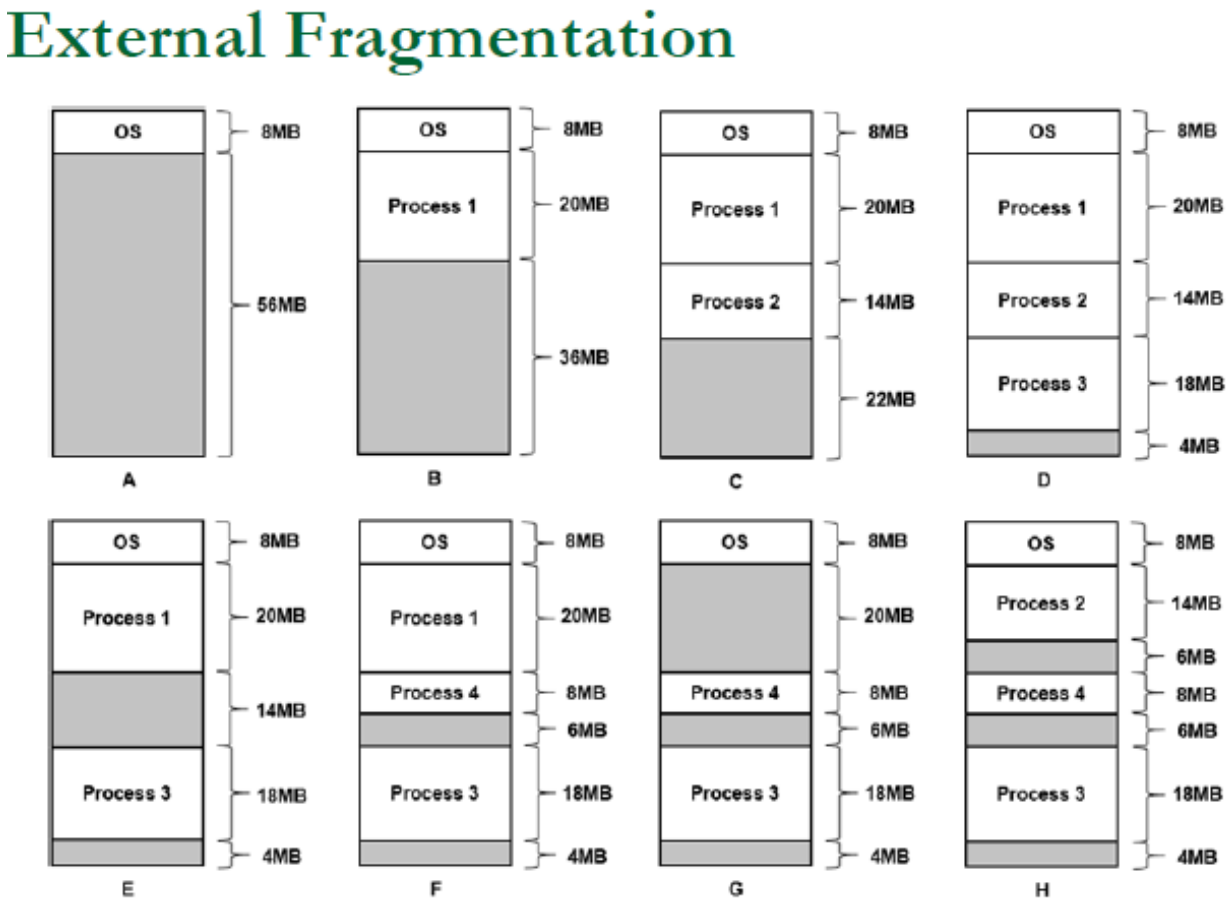
- Reduces internal fragmentation

Weaknesses

- If process is smaller than the smallest available partition, it may not utilize the partition space efficiently (**Internal fragmentation**).
- If process is larger than the largest partition in the system, program is required to redesign.
- There can be a process waiting in the queue of some partition while other partitions that are big enough to load it may remain empty.

Dynamic Partitioning

- Partitions memory dynamically based on the size of incoming processes.
- Incoming process is allocated exact amount of memory required.
- Strengths
  - No internal fragmentation
  - No need to redesign programs
  - Variable number of processes can use memory at a time
- Weaknesses
  - Dynamic memory allocation overhead
  - With the time, due to swapping ins and outs, memory may become more and more externally fragmented (**external fragmentation**)
- Solution
  - Periodically move the processes in memory to a one end leaving a large free space at the other end (**periodical compaction/ defragmentation**)
  - Processing overheads



Virtual Memory

- A facility that allows user programs to access memory from a logical point of view irrespective to the amount of memory physically available.
- Allows to run larger programs even if the system doesn't have enough memory to load them.
- Only a portion of a process is loaded into memory for execution while other portions are kept on disk.
- Two techniques used in virtual memory
    - Segmentation
    - Paging

Segmentation

- Similar to dynamic partition.
- Divides processes into variable-size **segments** (typically corresponds to program subroutines).
- Transfers segments from disk and places them in any available space of memory (no needed to be adjacent).
- Process can run if the segment that contains the next instruction to execute is in memory.
- Features
    - No internal fragmentation
    - External fragmentation possible but very less
    - System overheads

Paging

- Most popular virtual memory technology.
- Divides processes into small equal size **pages**.
- Partitions memory into page size blocks (**page frames**).
- Transfers pages from disk to page frames in memory.
- A process can run if the page that contains the next instruction to execute is in memory.
- When a page is no longer needed, moves it back to disk.

Features

- No internal fragmentation (exception possible)
- No external fragmentation

Example (consider 4 processes A, B , C and D)

Memory No.	Frame
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Memory No.	Frame
0	A0
1	A1
2	A2
3	
4	
5	
6	
7	
8	
9	

Memory No.	Frame
0	A0
1	A1
2	A2
3	B0
4	B1
5	
6	
7	
8	
9	

Memory No.	Frame
0	A0
1	A1
2	A2
3	B0
4	B1
5	C0
6	C1
7	C2
8	
9	

Memory No.	Frame
0	A0
1	A1
2	A2
3	
4	
5	C0
6	C1
7	C2
8	
9	

Memory No.	Frame
0	A0
1	A1
2	A2
3	D0
4	D1
5	C0
6	C1
7	C2
8	D2
9	

- When processor issues memory access request for a byte in a page of a process (3rd page of processor D), a **logical address** (2502) is issued relative to beginning of process.
- However, 2503rd byte of process D would not be at 2502 **physical address** of memory.

Address Translation

- Processor issues logical addresses (**Virtual address**) relative to the beginning of a process.

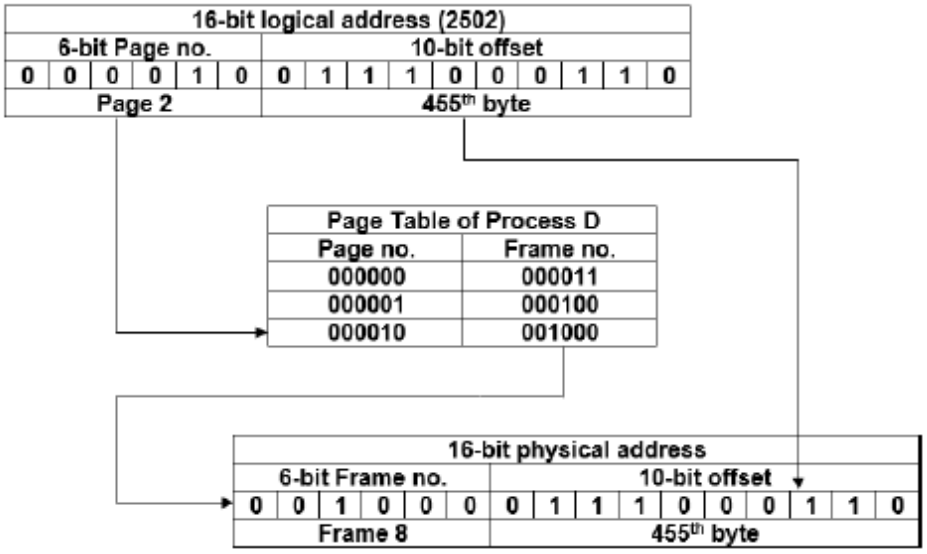
- Hardware called, **Memory Management Unit (MMU)** translates logical addresses into physical addresses (**real addresses**) of the underlying memory.
- For example, if page size is 1KB, logical address 2502 means a request to access 455th byte of page 2 of the process D.
  - Page 2 of process D means page D2 in frame number 8
  - So we need to find physical address of frame number 8

- If page size (thereby frame size) is 1KB and physical address is of 16 bits.

Memory		
Frame no.	Physical addresses	
	From	To
0	000000 0000000000 (0)	000000 1111111111 (1023)
1	000001 0000000000 (1024)	000001 1111111111 (2047)
2	000010 0000000000 (2048)	000010 1111111111 (3071)
3	000011 0000000000 (3072)	000011 1111111111 (4095)
4	000100 0000000000 (4096)	000100 1111111111 (5119)
5	000101 0000000000 (5120)	000101 1111111111 (6143)
6	000110 0000000000 (6144)	000110 1111111111 (7167)
7	000111 0000000000 (7168)	000111 1111111111 (8191)
8	001000 0000000000 (8192)	001000 1111111111 (9215)
9	001001 0000000000 (9216)	001001 1111111111 (10239)

- 6 high order bits of the address give **frame number** and 10 low order bits give **offset within the frame** (addresses of 0 to 1023 bytes within the frame)
- When processor issues a logical address, high order 6 bits of the address give **page number** and low order 10 bits give **offset within the page**.
- OS maintains a **page table** for each process.
- Page table keeps the track of frames that hold pages of a process in memory (frame numbers against page numbers)
- MMU uses page table of a process (process D) to find the physical address of frame that holds required page (page 2) in memory.

Example

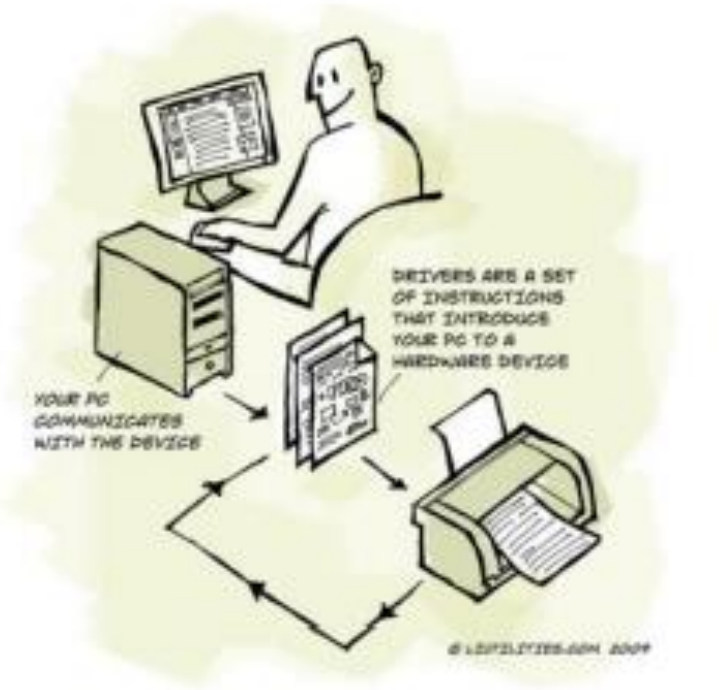


Input and output Device Management

- **Device driver**

Device driver is software. The computer communicates with peripheral devices through device drivers. A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without knowing the precise hardware details.

Device drivers depends on both the hardware and the operating system loaded in to the computer



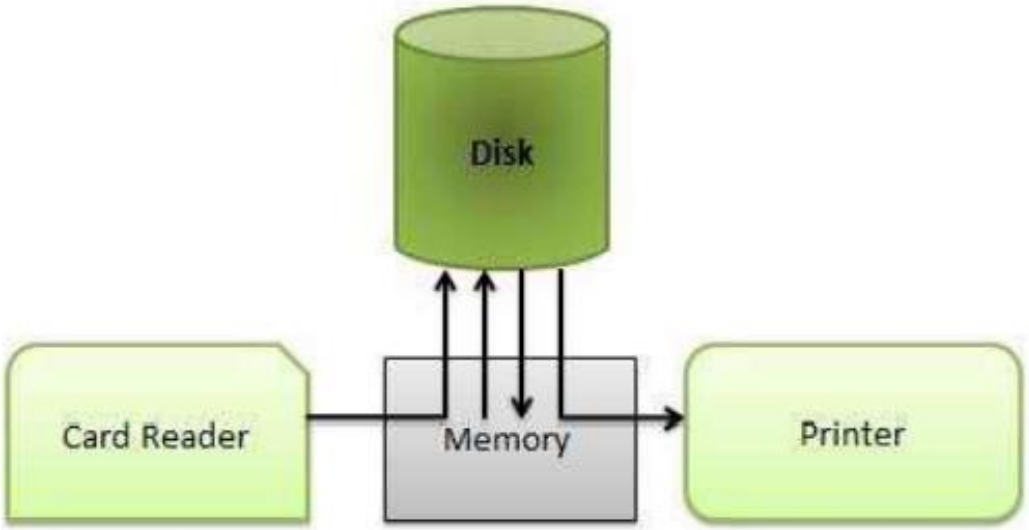
Spooling

Spooling is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.

An operating system does the following activities related to distributed environment:

- Handles I/O device data spooling as devices have different data access rates.
- Maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up.

- Maintains parallel computation because of spooling process as a computer can perform I/O in parallel fashion. It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task.



Advantages

- The spooling operation uses a disk as a very large buffer.
- Spooling is capable of overlapping I/O operation for one job with processor operations for another job

References

Teachers Guide – 2017

Chapter 7, W.Stallings, Operating System: Internals and Design Principles, 9th Edition, Pearson, 2017