



Lecture – 9  
Section-B

---

Theoretical concept of Unix  
Operating System



# Introduction

- Memory Management in UNIX OS
    - Swapping
    - Demand Paging
-



# Memory Management

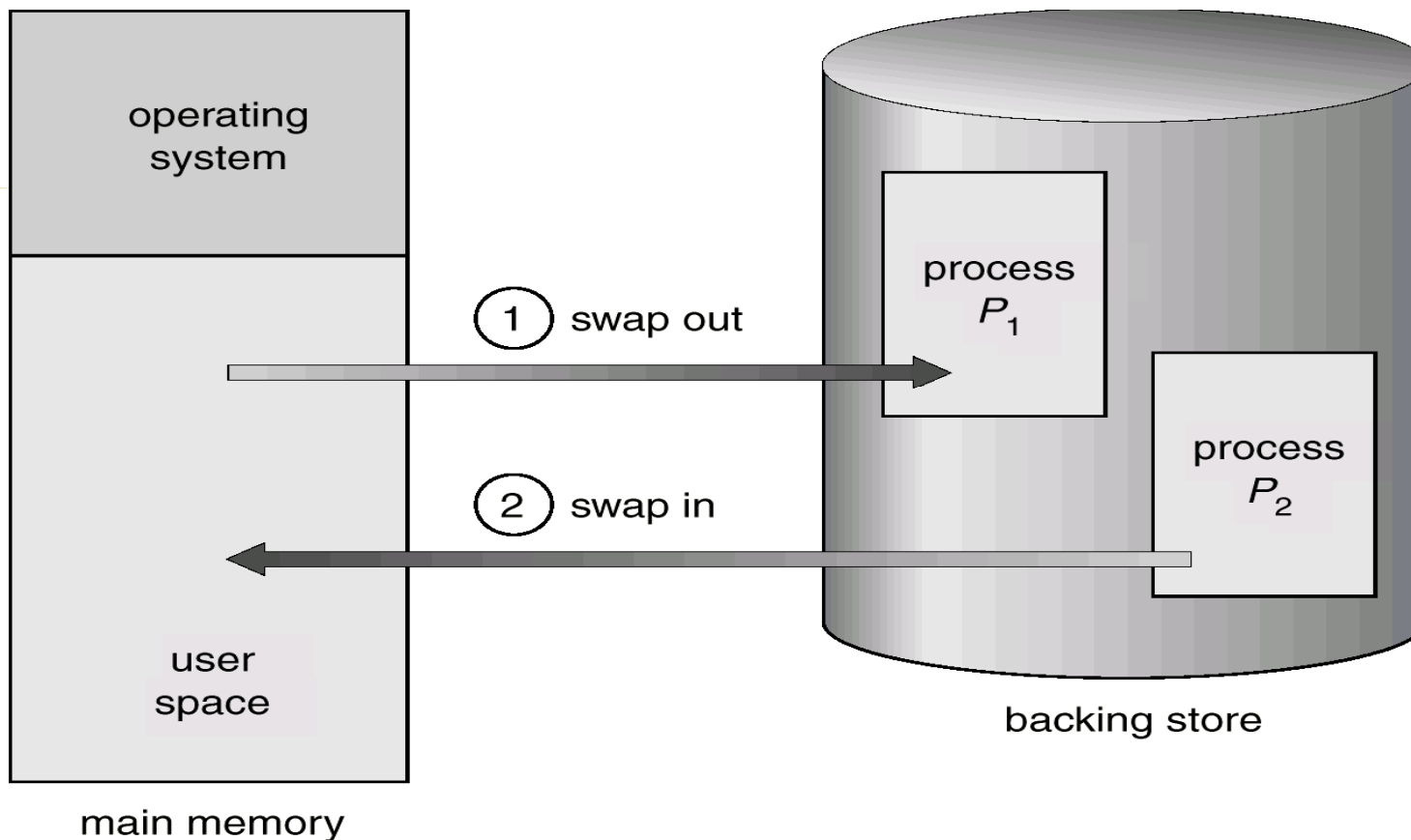
- **Memory management** is the act of managing computer memory.
- In its simpler forms, this involves providing ways to allocate portions of memory to programs at their request, and freeing it for reuse when no longer needed.

# UNIX Memory Management Policies

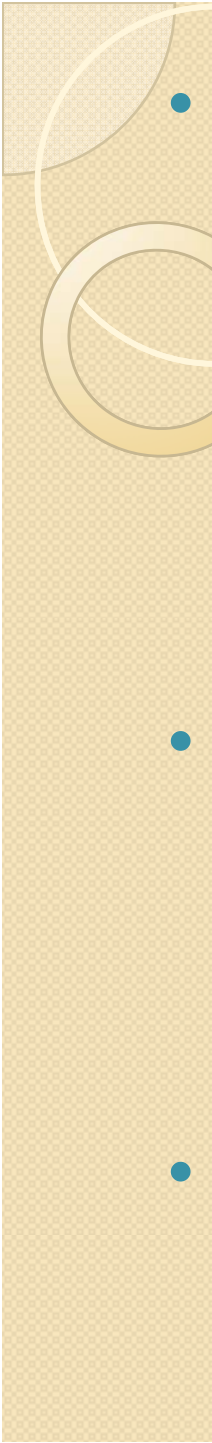
- **Swapping**
  - Easy to implement
  - Less system operating cost
- **Demand Paging**
  - Greater flexibility


# Swapping

• A process needs to be in memory to be executed. A process , however can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.



Swapping of two processes using a disk as a Backing Store

- 
- **For example:** assume a multiprogramming environment lets say a Unix programming environment, with a round robin CPU algorithm. When a quantum expires, the memory manager will start to swap out the process that just finished, and to swap in another process to the memory space that has been freed.
  - In the mean time the CPU scheduler will allocate a time slice to some other process in memory. When each process finishes its quantum, it will be swapped with another process.
  - ***The quantum must also be sufficiently large that reasonable amounts of computing are done between swaps.***

- 
- A variant of this swapping policy is used for priority based scheduling algorithms.
  - If a higher priority process arrives and wants service, the memory manager can swap out the lower priority process.
  - When the higher priority process finishes, the lower priority process can be swapped back in and continued. This variant of swapping can sometimes be called ***roll out, roll in.***
  - Swapping requires a backing store. The backing store is commonly a fast disk. It must be large enough to accommodate copies of all memory images for all users, and it must provide direct access to these

## Demand Paging

- Consider how an executable program might be loaded from disk into memory.
- **One option** – is to load the entire program in physical memory at program execution time.

However problem with this approach is we may not initially need the entire program in memory.

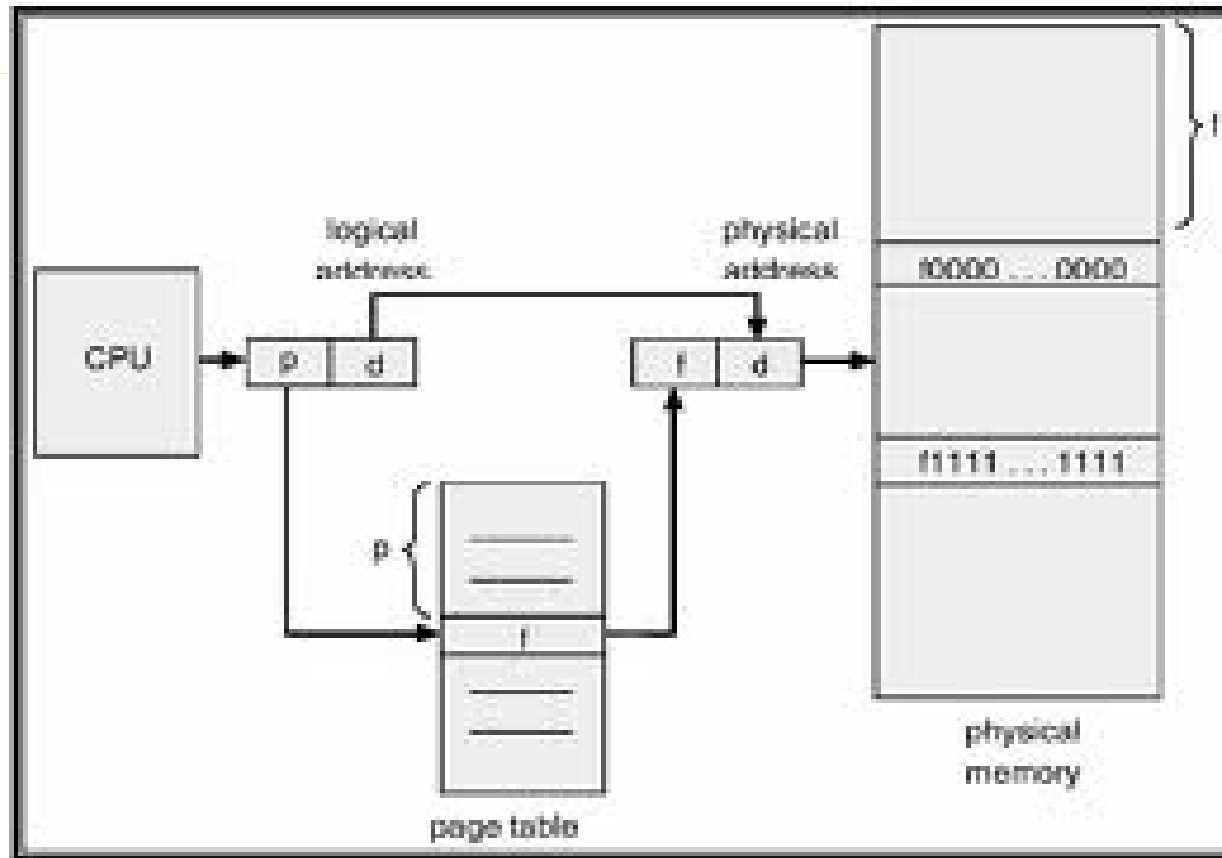
- Consider a program that starts with a list of available options from which the user is to select. Loading the entire program into memory results in loading the executable code for all options, regardless of whether an option is ultimately selected by the user or not.
- An **alternative strategy** is to initially load pages only as they are needed. This technique is known as Demand Paging.
- A Demand paging system is similar to a Paging System with Swapping, where processes reside in secondary memory ( Usually a disk.)

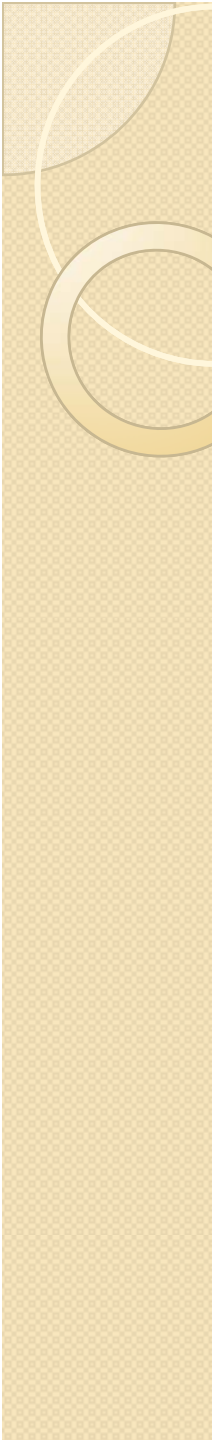


- **Paging System in brief :**

**Logical Address vs. Physical Address:** An address generated by the CPU is commonly referred to as ***Logical address*** .

Where as an address seen by the memory unit- that is, the one loaded into the memory address register of the memory – is commonly referred to as a ***physical address***.



- 
- Physical memory is broken into fixed-sized blocks called **frames**.
  - Logical memory is also broken into blocks of the same size called **pages**.
  - Set up a **page table** to translate logical to physical addresses.
  - When a process is to be executed, its pages are loaded into any available memory frames from the backing store. The backing store is divided into fixed blocks that are of the same size as the memory frames.
  - Every address generated by the CPU is divided into two parts : a **page number ( $p$ )** and a **page offset ( $d$ )**.
  - The Page number is used as an index into a page table. the page table contains the base address of each page in physical memory.
  - This base address is combined with the page offset to define the physical memory address that is sent to the memory unit

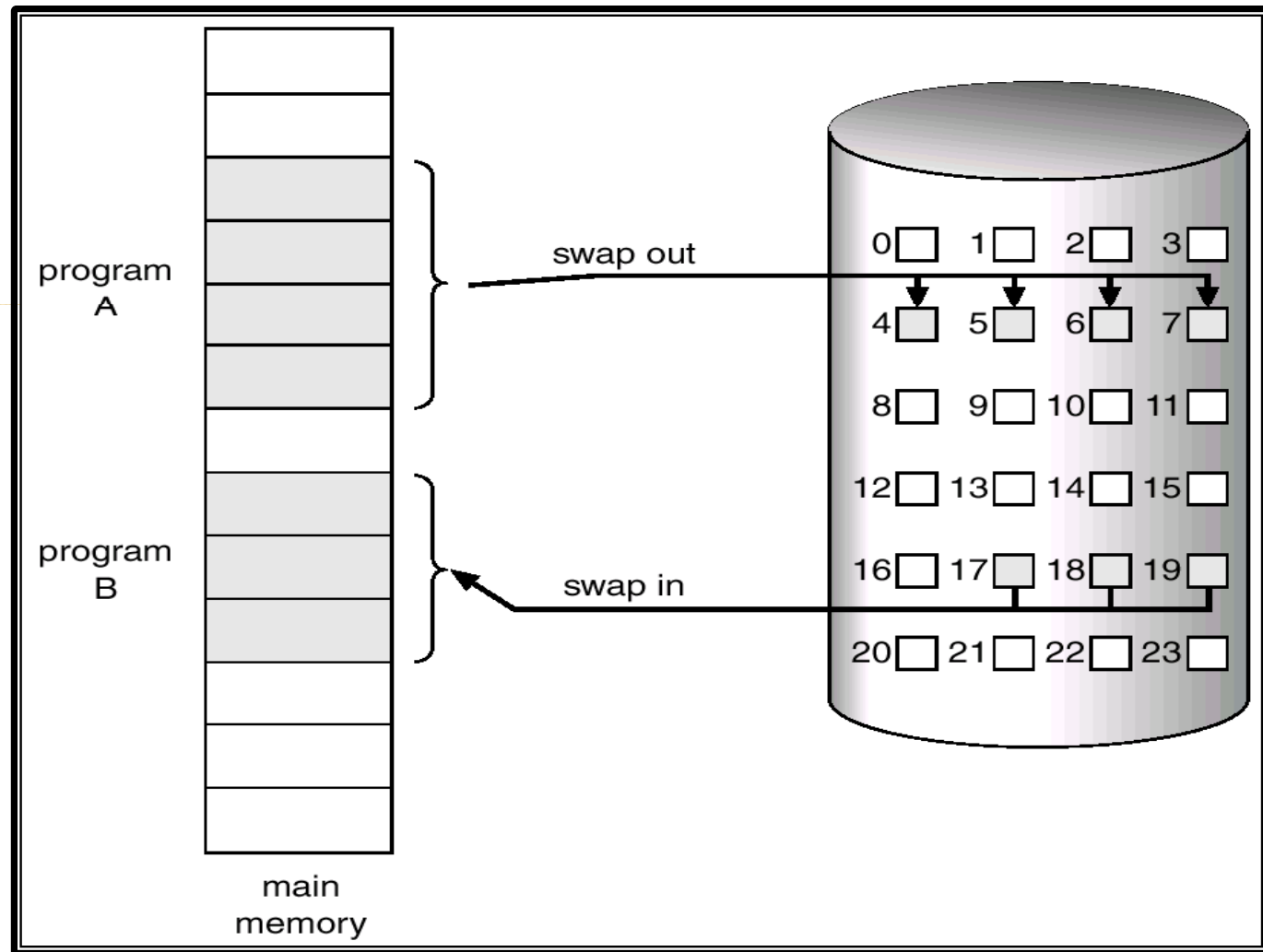
# Demand Paging

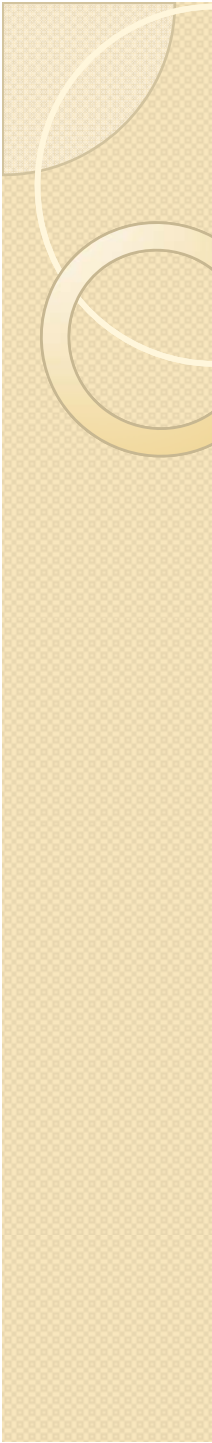
- Processes reside on secondary memory (which is usually a disk). when we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory, how ever we use a lazy swapper.
- A lazy swapper never swaps a page into memory unless that page will be needed.
- Since we are now viewing a process as a sequence of pages , rather than as one large contiguous address space, use of swap is technically incorrect. A swapper manipulates the entire processes, where as a **pager** is concerned with the individual pages of a process. We thus use a pager, rather than swapper, in connection with demand paging.

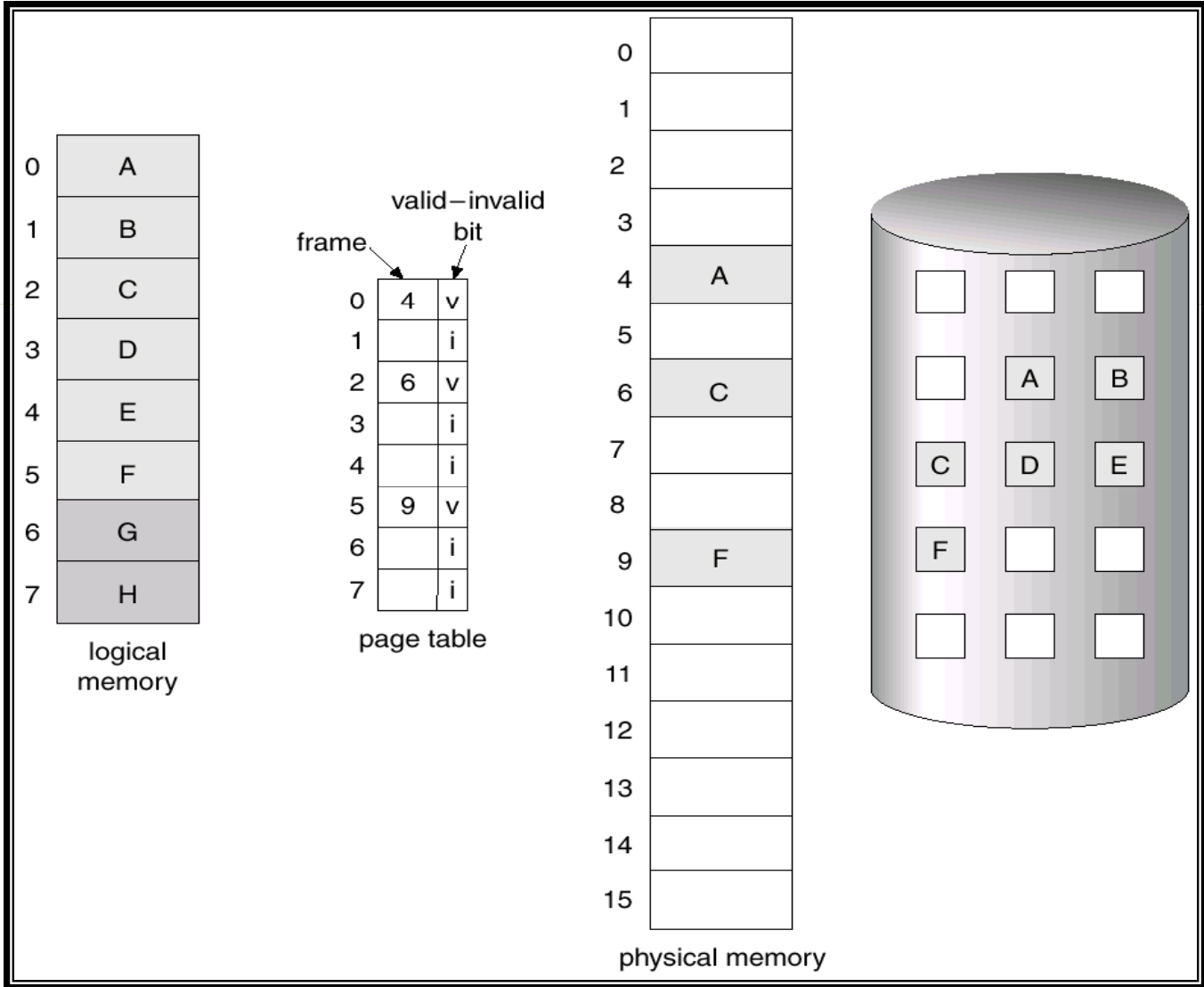


## **Basic Concept:**

- When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again.
- Instead of swapping in a whole process, the pager brings only those necessary pages into memory.
- Thus, it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed.



- 
- With this scheme, we need some form of hardware support to distinguish b/t those pages that are in memory and those pages that are on the disk.
  - The valid-invalid bit scheme can be used for this purpose.
  - With each page table entry a valid–invalid bit is associated
    - $v \Rightarrow$  in-memory,
    - $i \Rightarrow$  not-in-memory
  - Initially valid–invalid bit is set to  $i$  on all entries.





# Applications

## **Manual memory management**

- It can be easier for the programmer to understand exactly what is going on;
- Some manual memory managers perform better when there is a shortage of memory.

## **Automatic memory management**

- The programmer is freed to work on the actual problem;
- Module interfaces are cleaner;
- There are fewer memory management bugs;
- Memory management is often more efficient.





# Research

- Efficient dynamic memory management objectives include the following two points: the ability to quickly find and allocate free memory to ensure real-time; the need to reduce memory fragmentation, full use of the limited physical memory resources.
- Memory allocation means to determine the efficiency of memory management, which is the efficient use of limited memory key. Based on this, the paper RTDBS memory management conducted a study to improve the memory utilization of fully reflect real-time.