Memory Management in Recent Operating Systems

Project Members:

Aakriti banjara(05)
Saugat Khatri(24)
Prajwol Timalsina(57)
Regal Adhikari(64)

Presented To:

Mr. Sameer Tamrakar Sir
And CE-2020 classmates

Table of contents

01 Introduction

Memory management in an operating system serves as a foundational understanding of how the OS handles, allocates, and manages memory resources within a computer system.

Importance and Role in Modern Computing

- Resource Allocation and Optimization
- Multitasking and Process Management
- Virtual Memory Implementation
- Security and Memory Protection
- Performance and Speed Optimization
- Support for Large-scale Applications
- Dynamic Memory Management and Efficiency

02 Fundamentals of Memory Management

Memory Hierarchy Overview

Registers

Fastest memory unit that holds data that the CPU is actively processing.

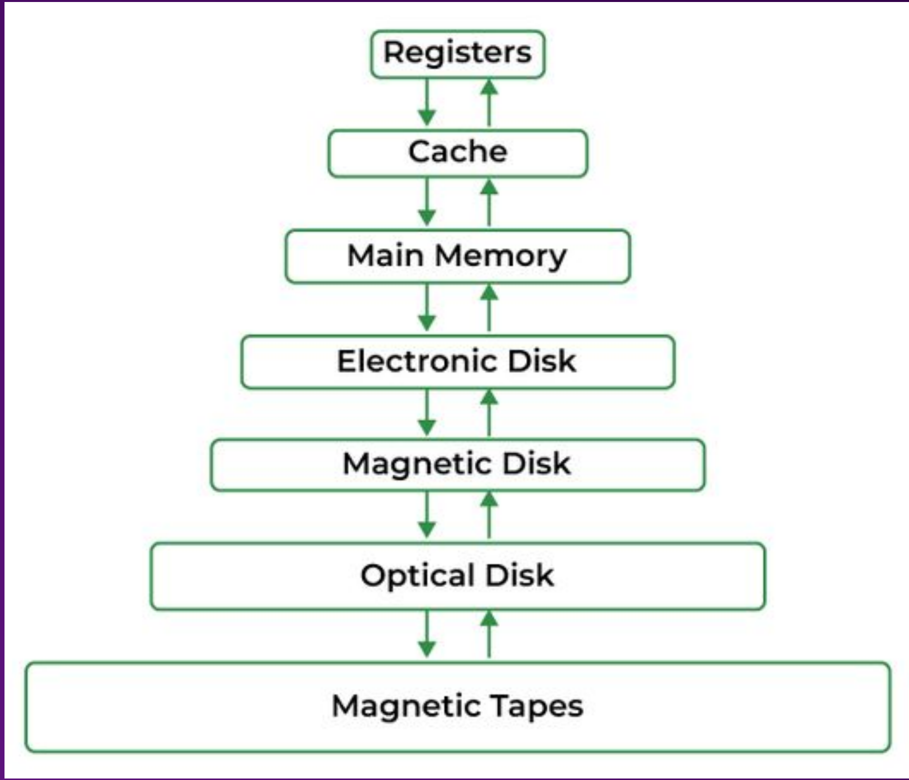
Cache Memory

Serves as a buffer, storing frequently accessed data and instructions.

Main Memory

Programs and data are stored in them while the CPU actively works on them.

Memory Hierarchy



Memory Management Unit(MMU)

- Translates virtual addresses generated by the CPU into physical addresses in the memory
- Enforces memory protection by controlling access to memory locations
- Facilitates the implementation of virtual memory, allowing systems to use more memory than physically available
- Manages the interaction between the CPU cache and main memory, optimizing the cache's efficiency
- Utilizes page tables to map virtual addresses to physical addresses

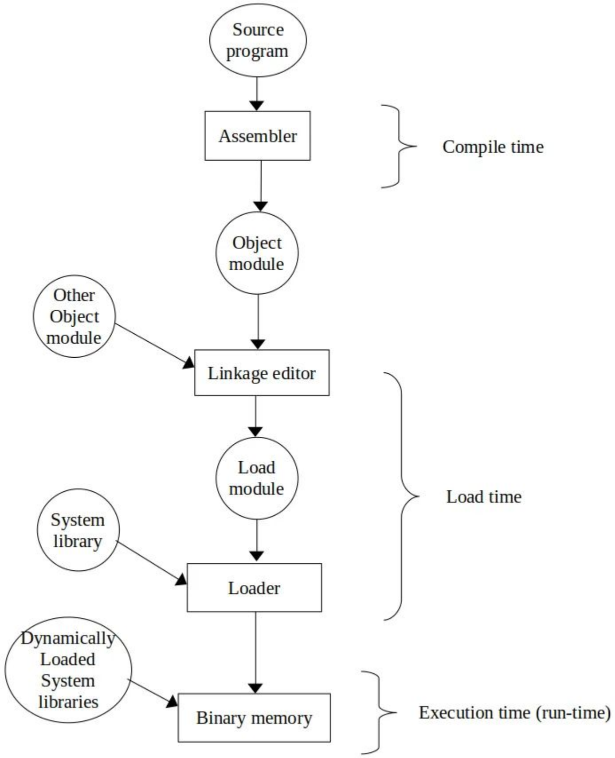
Address Binding

- Process of associating a symbolic or logical address with a physical address .

- Compile Time (Static) Binding:
  Absolute addresses for all the program's instructions and data generated during compile time .

- Load Time Binding:
  Final binding happens during the load time when the program is loaded into memory.

- Execution Time (Dynamic) Binding:
  Libraries or modules are linked to the program during execution rather than at compile or load time.



03 Memory Management Types and Schemes

Memory management schemes of Operating Systems

- Contiguous Allocation
- Non-contiguous Allocation
- Virtual Memory Allocation
- Hybrid Memory Allocation

Contiguous Memory Allocation

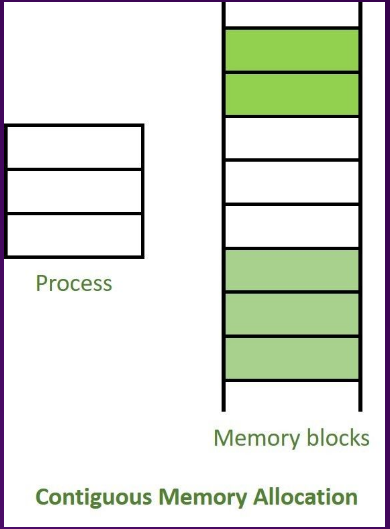
- Simplest memory management scheme - each process is assigned a contiguous block of memory space.
- The system maintains a table that records the base address and the limit of each process's memory block.

Advantages:

- Easy to implement ,fast access to memory.

Disadvantages:

- Suffers from external fragmentation



Non-Contiguous Memory Allocation

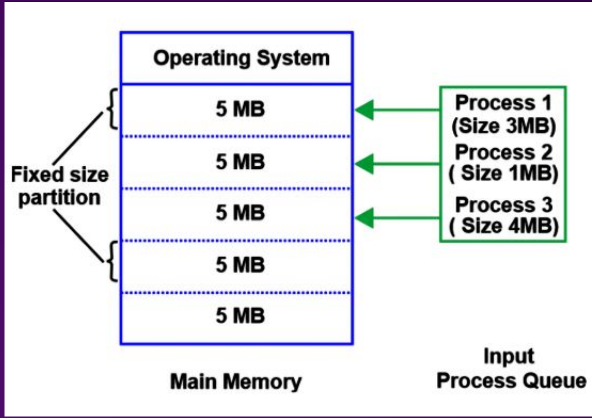
- Processes to be divided into smaller chunks of memory, called segments or pages that are stored in non-adjacent locations in memory.
- The system maintains a table that maps each segment or page of a process to its physical address in memory.

Advantages:

- eliminates external fragmentation
- Allows better utilization of memory

Disadvantages:

- Introduces complexity and overhead in managing the tables and accessing the memory



Types of Non-Contiguous Memory Management

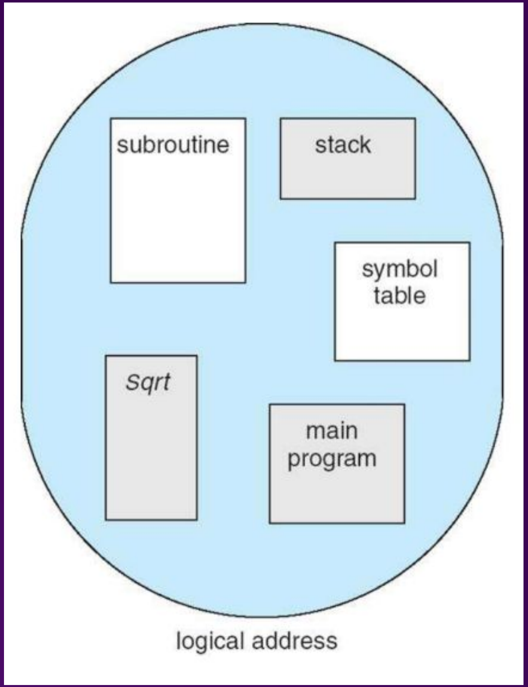
Segmentation

Paging

Virtual Memory

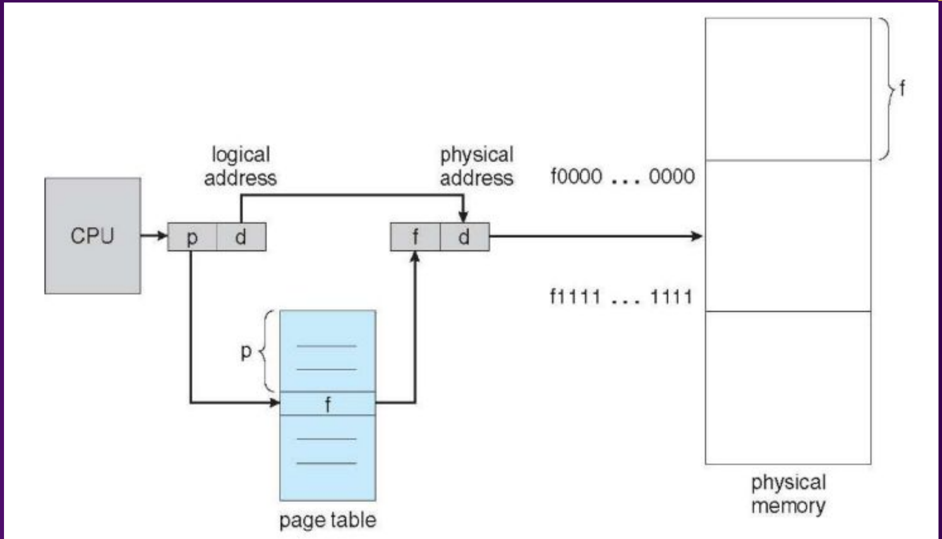
Segmentation

- Divides memory into segments of variable sizes based on logical units (e.g., code, data, stack).
- Minimizes internal fragmentation
- Offers flexibility but can lead to external fragmentation.



Paging

- Divide physical memory into fixed-sized blocks called frames
- Divide logical memory into blocks of same size called pages
- Eliminates external fragmentation
- Simplifies memory management but can lead to internal fragmentation.



Virtual Memory

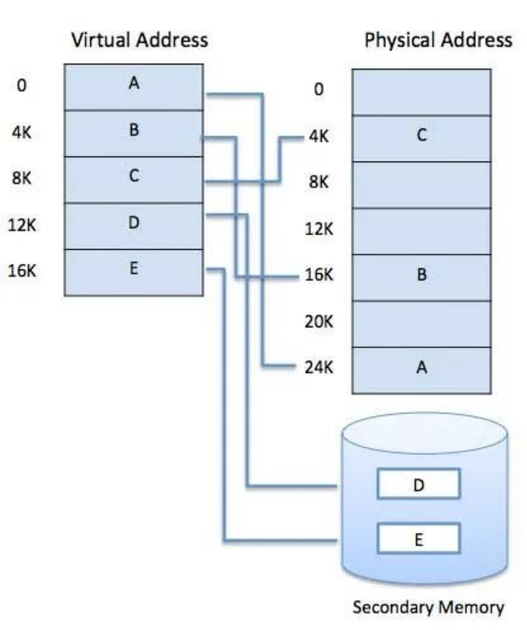
- Virtual memory extends the available memory space by using a secondary storage device, such as a disk, as an extension of the main memory.
- Divides the memory space into fixed-sized units, called pages or frames, and swaps them between the main memory and the disk as needed.

Advantages:

- Allows the system to run more processes than the main memory can hold
- Supports large and dynamic processes

Disadvantages:

- Introduces complexity and overhead in managing the tables and accessing the memory



Hybrid Scheme

- Combines two or more of the previous schemes to achieve a balance between simplicity, efficiency, and flexibility
- For example, use contiguous allocation for the kernel processes and non-contiguous allocation for the user processes

Advantages:

- Can adapt to different system needs and scenarios
- Higher overall performance

Disadvantages:

- May introduce compatibility or security issues
- Increases complexity and overhead

Address Translation and Page Faults

- ## Address Translation:
  Conversion of logical address into physical address.
  Memory management Unit(MMU) handles the address translation ,using page tables or mapping

- ## Page Faults:
  When a page fault happens, OS retrieves the required page from the disk into physical memory that involves disk I/O which can result in a temporary halt in the process's execution.

- ## Thrashing:
  Cpu performs swapping more than productive works.

04 Memory Management in Recent OS

Windows Memory Management

Memory Compression: Reduce the memory footprint of processes, utilizing algorithms to compress and store data more efficiently in RAM.

Dynamic Memory Allocation: Distribute memory resources among running processes or virtual machines.

Pagefile Optimization: Use when physical memory is insufficient.

Address Space Layout Randomization (ASLR): Windows implements ASLR to randomize memory addresses, enhancing security by making it harder for attackers to predict memory locations.

MacOS Memory Management

Unified Memory: Allow both CPU and GPU to access the same memory pool, enhancing performance for graphics-intensive tasks.

Memory Compression and Management: Similar to Windows, optimize memory usage and manages memory more efficiently, especially on systems with limited RAM.

Memory Pressure Management: Allocate and deallocate memory resources based on the system's workload.

File System Integration: Integration with Apple's file system (APFS) includes features for efficient memory handling, metadata management, and improved memory utilization.

Linux Memory Management

Transparent Huge Pages: Improve memory management for large memory allocations, enhancing performance for memory-intensive applications.

Kernel Same-page Merging (KSM): Identify identical memory pages across processes and merge them, reducing memory duplication and optimizing RAM usage.

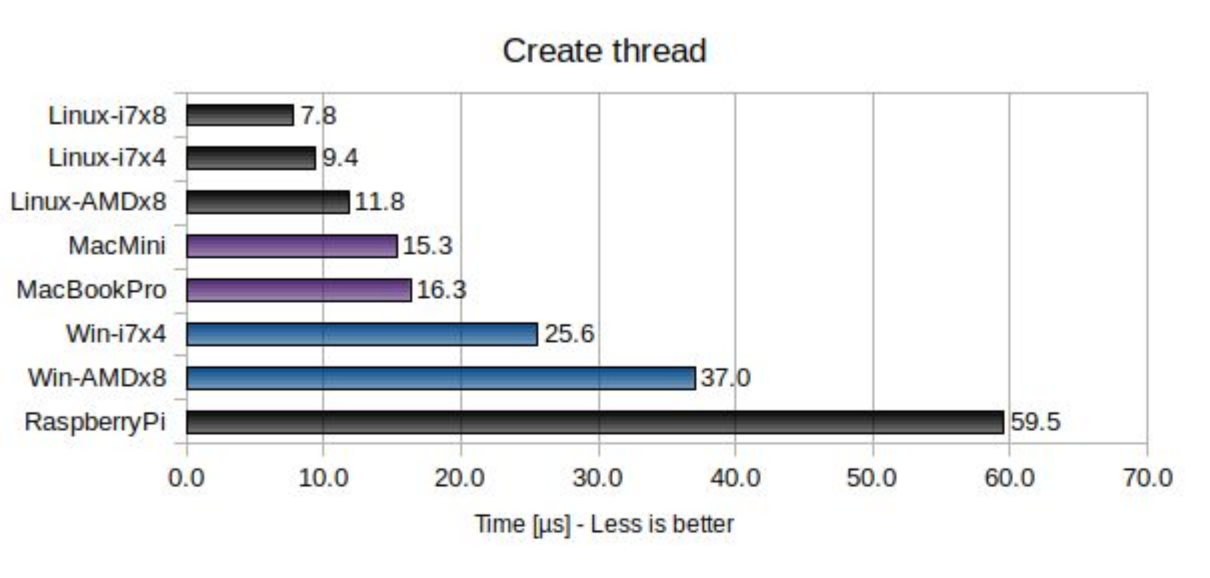
Control Groups (cgroups): Manage system resources, including memory, allowing administrators to allocate and limit memory usage for specific processes or groups.

Memory Ballooning: Similar to Windows, recent Linux versions implement memory ballooning techniques in virtualized environments to adjust memory allocation dynamically.

05 Performance Benchmarks and Improvements

Creating threads

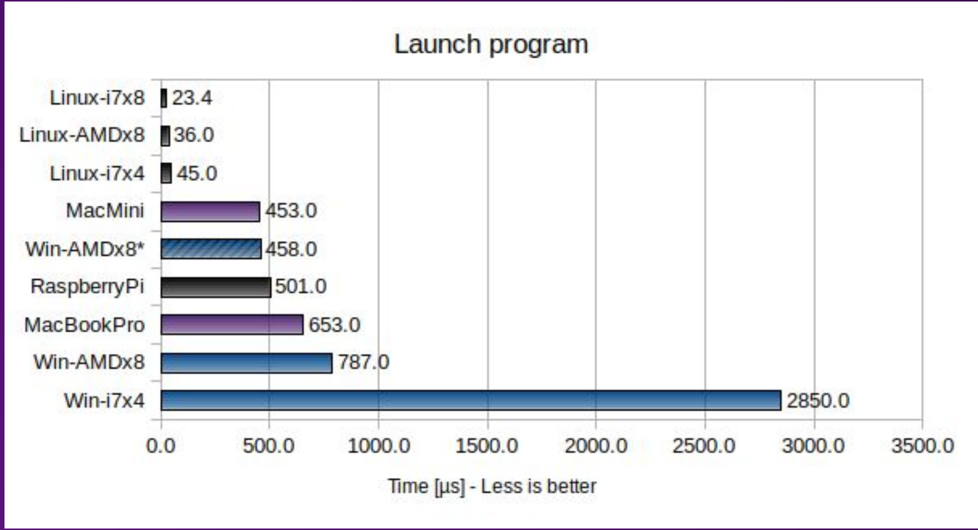
The graph below shows the time it takes for a single thread to start and terminate in different OS.



Apparently macOS is about twice as fast as Windows at creating threads, whereas Linux is about three times faster than Windows.

Launching programs

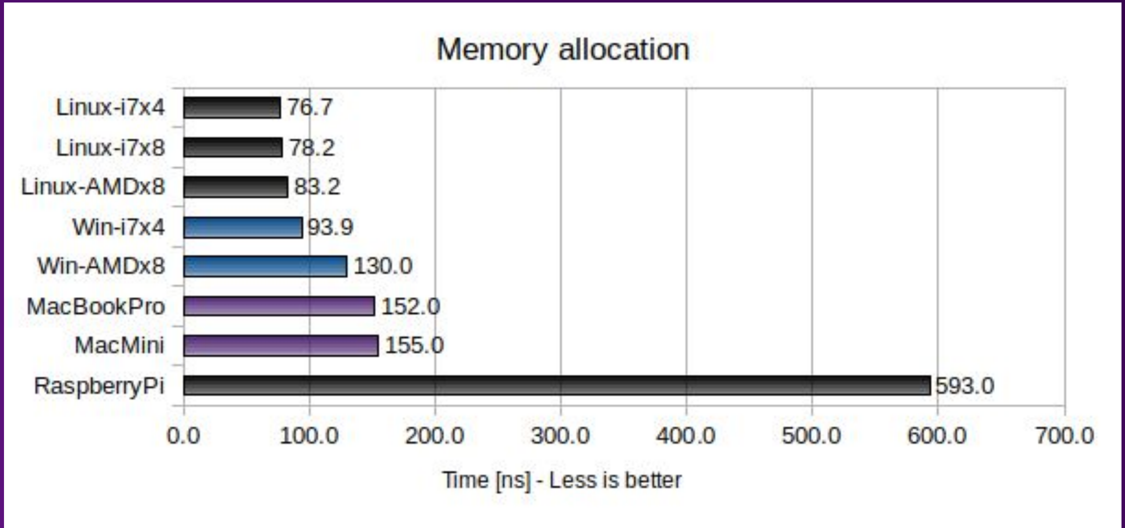
The graph below shows the time it takes for a program to be loaded and executed.



Here Linux is notably faster than both macOS (~10x faster) and Windows (>20x faster).

Allocating memory

The memory allocation performance was measured by allocating 1,000,000 small memory blocks (4-128 bytes in size) and then freeing them again.



Linux is slightly faster than both Windows and macOS

Thank You!

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**