5. Memory management

21 May 2022 19:00

- ## 5.1. Background

the two basic memory hardware mainly used in the computer are :

i. RAM : Random access memory.
ii. ROM : Read only memory.

- Memory is the central to the operation of a modern computer system.
- Memory consists of large array of words or bytes, each with its own address.
- The CPU fetches instructions from memory according to the value of the program counter.
- These instructions may cause additional loading from and storing to specific memory addresses.

What is address binding in the operating system?

The Address Binding refers to the mapping of computer instructions and data to physical memory locations. Both logical and physical addresses are used in computer memory. It assigns a physical memory region to a logical pointer by mapping a physical address to a logical address known as a virtual address.

Types of Address Binding in Operating System

1. **Compile Time Address Binding**
2. **Load Time Address Binding**
3. **Execution Time or Dynamic Address Binding**

Compile Time Address Binding

It is the first type of address binding. It occurs when the compiler is responsible for performing address binding, and the compiler interacts with the operating system to perform the address binding. In other words, when a program is executed, it allocates memory to the system code of the computer. The address binding assigns a logical address to the beginning of the memory segment to store the object code. Memory allocation is a long-term process and may only be modified by recompiling the program.

Load Time Address Binding

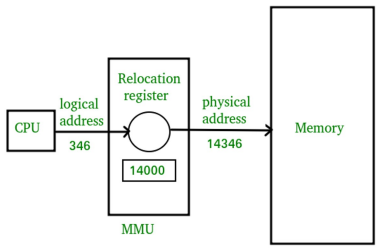
It is another type of address binding. It is done after loading the program in the memory, and it would be done by the operating system memory manager, i.e., loader. If memory allocation is specified when the program is assigned, no program in its compiled state may ever be transferred from one computer to another. Memory allocations in the executable code may already be in use by another program on the new system. In this case, the logical addresses of the program are not connected to physical addresses until it is applied and loaded into memory.

Execution Time or Dynamic Address Binding

Execution time address binding is the most popular type of binding for scripts that aren't compiled because it only applies to variables in the program. When a variable in a program is encountered during the processing of instructions in a script, the program seeks memory space for that variable. The memory would assign the space to that variable until the program sequence finished or unless a specific instruction within the script released the memory address connected to a variable.

Logical and Physical Address in Operating System

- **Logical Address** is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU.
- **Physical Address** identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address before they are used.



Dynamic relocation using a relocation register.

Parameter	LOGICAL ADDRESS	PHYSICAL ADDRESS
Basic	generated by CPU	location in a memory unit
Address Space	Logical Address Space is set of all logical addresses generated by CPU in reference to a program.	Physical Address is set of all physical addresses mapped to the corresponding logical addresses.
Visibility	User can view the logical address of a program.	User can never view physical address of program.
Generation	generated by the CPU	Computed by MMU
Access	The user can use the logical address to access the physical address.	The user can indirectly access physical address but not directly.
Editable	Logical address can be change.	Physical address will not change.
Also called	virtual address.	real address.

Swapping in Operating System

- Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes.
- It is used to improve main memory utilization. *
- In secondary memory, the place where the swapped-out process is stored is called swap space.
- The purpose of the swapping in operating system is to access the data present in the hard disk and bring it to RAM so that the application programs can use it. The thing to remember is that swapping is used only when data is not present in RAM.
- Although the process of swapping affects the performance of the system, it helps to run larger and more than one process. This is the reason why swapping is also referred to as memory compaction.

The concept of swapping has divided into two more concepts: Swap-in and Swap-out.

- Swap-out is a method of removing a process from RAM and adding it to the hard disk.
- Swap-in is a method of removing a program from a hard disk and putting it back into the main memory or RAM.

Example: Suppose the user process's size is 2048KB and is a standard hard disk where swapping has a data transfer rate of 1Mbps. Now we will calculate how long it will take to transfer from main memory to secondary memory.

User process size is 2048Kb
Data transfer rate is 1Mbps = 1024 kbps
Time = process size / transfer rate
= 2048 / 1024
= 2 seconds
= 2000 milliseconds
Now taking swap-in and swap-out time, the process will take 4000 milliseconds.

Advantages of Swapping

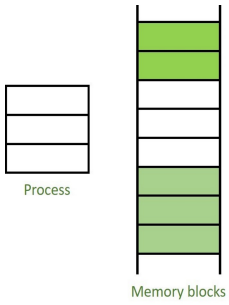
1. It helps the CPU to manage multiple processes within a single main memory.
2. It helps to create and use virtual memory.
3. Swapping allows the CPU to perform multiple tasks simultaneously. Therefore, processes do not have to wait very long before they are executed.
4. It improves the main memory utilization.

Disadvantages of Swapping

1. If the computer system loses power, the user may lose all information related to the program in case of substantial swapping activity.
2. If the swapping algorithm is not good, the composite method can increase the number of Page Fault and decrease the overall processing performance.

1. Contiguous Memory Allocation :

Contiguous memory allocation is basically a method in which a single contiguous section/part of memory is allocated to a process or file needing it. Because of this all the available memory space resides at the same place together, which means that the freely/unused available memory partitions are not distributed in a random fashion here and there across the whole memory space.



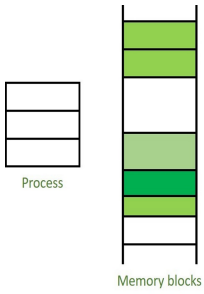
Contiguous Memory Allocation

The main memory is a combination of two main portions- one for the operating system and other for the user program. We can implement/achieve contiguous memory allocation by dividing the memory partitions into fixed size partitions.

2. Non-Contiguous Memory Allocation :

Non-Contiguous memory allocation is basically a method on the contrary to contiguous allocation method, allocates the memory space present in different locations to the process as per it's requirements. As all the available memory space is in a distributed pattern so the freely available memory space is also scattered here and there.

This technique of memory allocation helps to reduce the wastage of memory, which eventually gives rise to Internal and external fragmentation.



Noncontiguous Memory Allocation

Difference between Contiguous and Non-contiguous Memory Allocation :

S.NO.	Contiguous Memory Allocation	Non-Contiguous Memory Allocation
1.	Contiguous memory allocation allocates consecutive blocks of memory to a file/process.	Non-Contiguous memory allocation allocates separate blocks of memory to a file/process.
2.	Faster in Execution.	Slower in Execution.
3.	It is easier for the OS to control.	It is difficult for the OS to control.
4.	Overhead is minimum as not much address translations are there while executing a process.	More Overheads are there as there are more address translations.
5.	Both Internal fragmentation and external fragmentation	External fragmentation occurs in Non-Contiguous memory

	occurs in Contiguous memory allocation method.	allocation method.
6.	It includes single partition allocation and multi-partition allocation.	It includes paging and segmentation.
7.	Wastage of memory is there.	No memory wastage is there.
8.	In contiguous memory allocation, swapped-in processes are arranged in the originally allocated space.	In non-contiguous memory allocation, swapped-in processes can be arranged in any place in the memory.

What is Fragmentation?

- Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented.
- Fragmentation. As processes are loaded and removed from memory, the free memory space is broken into little pieces.

Causes of Fragmentation

User processes are loaded and unloaded from the main memory, and processes are kept in memory blocks in the main memory. Many spaces remain after process loading and swapping that another process cannot load due to their size. Main memory is available, but its space is insufficient to load another process because of the dynamical allocation of main memory processes.

Types of Fragmentation

There are mainly two types of fragmentation in the operating system. These are as follows:

1. **Internal Fragmentation**
2. **External Fragmentation**

Internal Fragmentation

When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes **internal** fragmentation.

How to avoid internal fragmentation?

The problem of internal fragmentation may arise due to the fixed sizes of the memory blocks. It may be solved by assigning space to the process via dynamic partitioning. Dynamic partitioning allocates only the amount of space requested by the process. As a result, there is no internal fragmentation.

External Fragmentation

External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as **external** fragmentation.

How to remove external fragmentation?

This problem occurs when you allocate RAM to processes continuously. It is done in paging and segmentation, where memory is allocated to processes non-contiguously. As a result, if you remove this condition, external fragmentation may be decreased.

Advantages and disadvantages of fragmentation

There are various advantages and disadvantages of fragmentation. Some of them are as follows:

Advantages

Fast Data Writes

Data write in a system that supports data fragmentation may be faster than reorganizing data storage to enable contiguous data writes.

Fewer Failures

If there is insufficient sequential space in a system that does not support fragmentation, the write will fail.

Storage Optimization

A fragmented system might potentially make better use of a storage device by utilizing every available storage block.

Disadvantages

Need for regular defragmentation

A more fragmented storage device's performance will degrade with time, necessitating the requirement for time-consuming defragmentation operations.

Slower Read Times

The time it takes to read a non-sequential file might increase as a storage device becomes more fragmented.

What is Page Fault in Operating System?

- Page faults dominate more like an **error**. A page fault will happen if a program tries to access a piece of memory that does not exist in physical memory (main memory).
- A page fault trap occurs if the requested page is not loaded into memory. The page fault primarily causes an exception, which is used to notify the operating system to retrieve the **"pages"** from virtual memory to continue operation. Once all of the data has been placed into physical memory, the program resumes normal operation.

1. The computer's hardware track to the kernel and the program counter is often saved on the stack. The CPU registers hold information about the current state of instruction.
2. An assembly program is started, which saves the general registers and other volatile data to prevent the Operating system from destroying it.

Page Fault Handling

A Page Fault happens when you access a page that has been marked as invalid. The paging hardware would notice that the invalid bit is set while translating the address across the page table, which will cause an operating system trap. The trap is caused primarily by the OS's failure to load the needed page into memory.

Now, let's understand the procedure of page fault handling in the OS:

1. Firstly, an internal table for this process to assess whether the reference was valid or invalid memory access.
2. If the reference becomes invalid, the system process would be terminated. Otherwise, the page will be paged in.
3. After that, the free-frame list finds the free frame in the system.
4. Now, the disk operation would be scheduled to get the required page from the disk.
5. When the I/O operation is completed, the process's page table will be updated with a new frame number, and the invalid bit will be changed. Now, it is a valid page reference.
6. If any page fault is found, restart these steps from starting.

Page Fault Terminology

There are various page fault terminologies in the operating system. Some terminologies of page fault are as follows:

1. Page Hit
When the CPU attempts to obtain a needed page from main memory and the page exists in **main memory (RAM)**, it is referred to as a **"PAGE HIT"**.

2. Page Miss
If the needed page has not existed in the **main memory (RAM)**, it is known as **"PAGE MISS"**.

3. Page Fault Time
The time it takes to get a page from secondary memory and recover it from the main memory after loading the required page is known as **"PAGE FAULT TIME"**.

4. Page Fault Delay
The rate at which threads locate page faults in memory is referred to as the **"PAGE FAULT RATE"**. The page fault rate is measured per second.

5. Hard Page Fault
If a required page exists in the hard disk's page file, it is referred to as a **"HARD PAGE FAULT"**.

6. Soft Page Fault
If a required page is not located on the hard disk but is found somewhere else in memory, it is referred to as a **"SOFT PAGE FAULT"**.

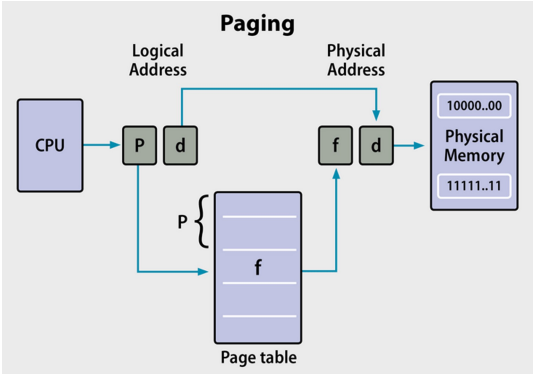
7. Minor Page Fault
If a process needs data and that data exists in memory but is being allotted to another process at the same moment, it is referred to as a **"MINOR PAGE FAULT"**.

What is Paging?

the memory management function called *paging* specifies storage locations to the CPU as additional memory, called virtual memory. The CPU cannot directly access storage disk, so the MMU emulates memory by mapping pages to frames that are in RAM.

Before we launch into a more detailed explanation of pages and frames, let's define some technical terms.

- **Page:** A fixed-length contiguous block of virtual memory residing on disk.
- **Frame:** A fixed-length contiguous block located in RAM; whose sizing is identical to pages.
- **Physical memory:** The computer's random access memory (RAM), typically contained in DIMM cards attached to the computer's motherboard.
- **Virtual memory:** Virtual memory is a portion of an HDD or SSD that is reserved to emulate RAM. The MMU serves up virtual memory from disk to the CPU to reduce the workload on physical memory.
- **Virtual address:** The CPU generates a virtual address for each active process. The MMU maps the virtual address to a physical location in RAM and passes the address to the bus. A virtual address space is the range of virtual addresses under CPU control.
- **Physical address:** The physical address is a location in RAM. The physical address space is the set of all physical addresses corresponding to the CPU's virtual addresses. A physical address space is the range of physical addresses under MMU control.



The Paging Process

A page table stores the definition of each page. When an active process requests data, the MMU retrieves corresponding pages into frames located in physical memory for faster processing. The process is called paging.

The MMU uses page tables to translate virtual addresses to physical ones. Each table entry indicates where a page is located: in RAM or on disk as virtual memory. Tables may have a single or multi-level page table such as different tables for applications and segments.

However, constant table lookups can slow down the MMU. A memory cache called the Translation Lookaside Buffer (TLB) stores recent translations of virtual to physical addresses for rapid retrieval. Many systems have multiple TLBs, which may reside at different locations, including between the CPU and RAM, or between multiple page table levels.

Different frame sizes are available for data sets with larger or smaller pages and matching-sized frames. 4KB to 2MB are common sizes, and GB-sized frames are available in high-performance servers.

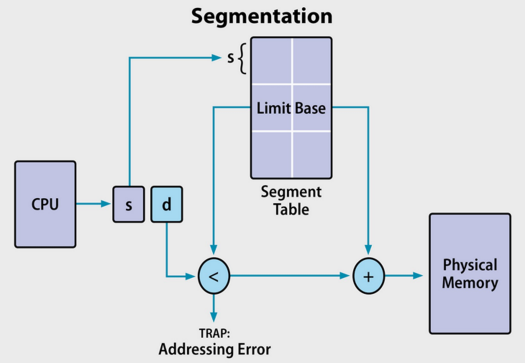
An issue called hidden fragmentation used to be a problem in older Windows deployments (95, 98, and Me). The problem was internal (or hidden) fragmentation. Unlike the serious external fragmentation of segmenting, internal fragmentation occurred if every frame is not the exact size of the page size. However, this is not an issue in modern Windows OS.

What is Segmentation?

The process known as *segmentation* is a virtual process that creates address spaces of various sizes in a computer system,

called segments. Each segment is a different virtual address space that directly corresponds to process objects.

When a process executes, segmentation assigns related data into segments for faster processing. The segmentation function maintains a segment table that includes physical addresses of the segment, size, and other data.



The Segmentation Process

Each segment stores the processes primary function, data structures, and utilities. The CPU keeps a segment map table for every process and memory blocks, along with segment identification and memory locations.

The CPU generates virtual addresses for running processes. Segmentation translates the CPU-generated virtual addresses into physical addresses that refer to a unique physical memory location. The translation is not strictly one-to-one: different virtual addresses can map to the same physical address.

The Challenge of Fragmentation

Although segmentation is a high-speed and highly secure memory management function, external fragmentation proved to be an insurmountable challenge. Segmentation causes external fragmentation to the point that modern x86-64 servers treat it as a legacy application, and only support it for backwards compatibility.

External fragmentation occurs when unusable memory is located outside of allocated memory blocks. The issue is that the system may have enough memory to satisfy process request, but the available memory is not in a contiguous location. In time, the fragmentation worsens and significantly slows the segmentation process.

Segmented Paging

Some modern computers use a function called segmented paging. Main memory is divided into variably-sized segments, which are then divided into smaller fixed-size pages on disk. Each segment contains a page table, and there are multiple page tables per process.

Each of the tables contains information on every segment page, while the segment table has information about every segment. Segment tables are mapped to page tables, and page tables are mapped to individual pages within a segment.

Advantages include less memory usage, more flexibility on page sizes, simplified memory allocation, and an additional level of data access security over paging. The process does not cause external fragmentation.

Paging and Segmentation: Advantages and Disadvantages

Paging Advantages

- On the programmer level, paging is a transparent function and does not require intervention.
- No external fragmentation.
- No internal fragmentation on updated OS’s.
- Frames do not have to be contiguous.

Paging Disadvantages

- Paging causes internal fragmentation on older systems.
- Longer memory lookup times than segmentation; remedy with TLB memory caches.

Segmentation Advantages

- No internal fragmentation.
- Segment tables consumes less space compared to page tables.
- Average segment sizes are larger than most page sizes, which allows segments to store more process data.
- Less processing overhead.
- Simpler to relocate segments than to relocate contiguous address spaces on disk.
- Segment tables are smaller than page tables, and takes up less memory.

Segmentation Disadvantages

- Uses legacy technology in x86-64 servers.
- Linux only supports segmentation in 80×86 microprocessors: states that paging simplifies memory management by using the same set of linear addresses.
- Porting Linux to different architectures is problematic because of limited segmentation support.
- Requires programmer intervention.
- Subject to serious external fragmentation.

Key Differences: Paging and Segmentation

Size:

- **Paging:** Fixed block size for pages and frames. Computer hardware determines page/frame sizes.
- **Segmentation:** Variable size segments are user-specified.

Fragmentation:

- **Paging:** Older systems were subject to internal fragmentation by not allocating entire pages to memory. Modern OS's no longer have this problem.
- **Segmentation:** Segmentation leads to external fragmentation.

Tables:

- **Paging:** Page tables direct the MMU to page location and status. This is a slower process than segmentation tables, but TLB memory cache accelerates it.
- **Segmentation:** Segmentation tables contain segment ID and information, and are faster than direct paging table lookups.

Availability:

- **Paging:** Widely available on CPUs and as MMU chips.
- **Segmentation:** Windows servers may support backwards compatibility, while Linux has very limited support.

- **Virtual Memory**
  Virtual memory is a memory management technique that can be implemented using both hardware and software. As the name indicates, it adds virtual memory to available memory, so that your system will appear to have more memory than what actually exists. Virtual memory is a layer of memory addresses (virtual addresses) that map to physical addresses. The memory addresses used by a program is the virtual addresses. These virtual address spaces and the assignment of real memory to the virtual memory are managed by the operating system.
- Demand Paging
  Demand paging is a type of swapping done in virtual memory systems. In demand paging, the data is not copied from the disk to the RAM until they are needed or being demanded by some program. The data will not be copied when the data is already available on the memory. This is otherwise called a lazy evaluation because only the demanded pages of memory are being swapped from the secondary storage (disk space) to the main memory.

  Demand Paging Working
  The demand paging working is based on a page table implementation. The page table maps logical memory to physical memory. The page table uses a bitwise operator to mark if a page is valid or invalid. A valid page is one that currently resides in main memory. An invalid page can be defined as the one that currently resides in secondary memory. When a process tries to access a page, the following will happen.
  1) Attempt to access the page
  2) The page is valid. Page processing instruction continues as normal.
  3) If the page is an invalid one, then a page-fault trap occurs.

4) The memory reference is then checked to determine if it is a valid reference to a location on secondary memory or not. If not, the process is terminated (illegal memory access). Otherwise, the required page is paged in.
5) Now the disk operation to read the desired page into main memory is scheduled.
6) Finally, the instruction that was interrupted by the operating system trap is restarted.
If you need any further assistance please contact our support department.