

Sultan Qaboos University

College of Science

Department of Computer Science

COMP4501 – Operating Systems

Virtual Memory

Research Report (Final draft)

**Coordinator:** Aya Al Balushi 135131

**Member1:** Khadija Al Bulushi 133556

**Member2:** Quds Al Breiki 133343

**Member3:** Maysoon Ambu Saidi 135352

**Date:** 9 December 2023

**Contents**

[Overview 3](#_heading=h.gjdgxs)

[Objectives 3](#_heading=)

[Concept 3](#_heading=h.79edc4517bil)

[Design History 4](#_heading=h.kq764622qin3)

[Original Design 5](#_heading=h.d4g1ug7ezy9a)

[Progress in Development 6](#_heading=h.2et92p0)

[Current Trends 6](#_heading=h.tyjcwt)

[Importance to OS 6](#_heading=h.3dy6vkm)

[Challenges 7](#_heading=h.c2wikuesgf98)

[Prospect Future Design Plans 7](#_heading=h.ewu0csgn89vl)

[Program concept 7](#_heading=h.clq9v3o0zl4b)

[Practical part 8](#_heading=h.pftnfidnyzi1)

[References 11](#_heading=h.gkjb1d3khw67)

# **Overview**

Virtual memory is a memory management technique used in operating systems to manage the storage space to maintain the speed and efficiency of the system during execution. This report explored the Virtual Memory technique, starting with its concept and how it works. Then, provides an insight into the evolution of design over time, the very first innovation and creation of the technique design, the ongoing improvements and adaptation to modern computing requirements, and its current trends. Next, it lists some of the advantages of virtual memory to the operating systems to enable the effective execution of diverse programs, finishing up with Future Design Plans and strategies.

# **Objectives**

* Explaining the concept of virtual memory and its design.
* Describing the benefits of virtual memory to the operating system.
* Understanding the design history and its very first design of virtual memory.
* Exploring the progress of development, current trends, and future plans for virtual memory.

# **Concept**

Virtual Memory is a storage allocation scheme used by modern operating systems in which it provides an illusion of a larger and more capable main memory (RAM) than is physically available by addressing the secondary memory (Hard disk). It is a technique that uses both the computer's hardware and software, where it takes program addresses and maps them to physical addresses (RAM addresses) in computer memory. Most operating systems have their method to configure the amount of disk space to allocate for virtual memory use. In general, hard disk virtual memory is relatively slower than the physical memory. When a program needs to be stored in memory but is not frequently used, this block of data is moved to a hard disk virtual memory, while leaving the physical memory free for data that needs to be read from or written too often. This implies that a process can be swapped in and out of the main memory, occupying different places in the main memory at different times during the execution.  The mapping technique helps to elevate the system's performance, efficiency, and flexibility.

There are different techniques used in virtual memory, here are some of the common techniques:

1. **Paging:**

Memory is split up into small, equal-sized sections called pages (or page frames). A single application may occupy multiple pages, which are not necessarily contiguous. Each application program has its own view of memory, known as logical memory. There are page table records where the different pages of a program are located in physical memory. Unused pages may be paged out to a swap file on hard disc to make room for others, and when are needed again, they are paged in. In paging, when memory is low, excessive swapping can lead to disc threshing and degraded performance.

1. **Segmentation:**

Program segments correspond to blocks of program code such as procedures or functions in memory, where segments vary in size. The operating system knows the start and size of each segment in physical memory, each segment is atomic, either the whole segment is in RAM, or none of the segment is in RAM. Segments are swapped between hard disc and main memory as needed where each segment in memory can only be replaced by a segment of the same size or smaller. Segmentation can result in memory fragmentation; a lot of small segments with gaps in between. In addition, large segments may only be allowed into memory sometimes.

# **Design History**

**EARLY VIRTUAL MACHINES**

Early efforts in developing virtual machines were a natural extension of the progress in multiprogramming, with the ongoing objective of securely distributing a physical machine's resources among several concurrent processes. It was possible to create the illusion of an independent physical machine running a complete operating system.

1. **M44/44X**

In 1964, IBM's M44 project extended the concepts of multiprogramming and introduced early virtual memory to enhance process isolation and scheduling within the privileged kernel.

1. **Cambridge Monitor System**

In the mid-1960s, the CP-40/CMS project on a modified System/360 (model 40) introduced the concept of virtual machines. This innovation allowed running software designed for specific hardware without modification. It also improved isolation and performance through privilege separation, paged memory, and simulated devices.

1. **VM/370**

In the early 1970s, IBM introduced VM/370 on System/370 hardware, which featured virtual memory capabilities.

**MODERN VIRTUAL MACHINES**

In the 1980s and 1990s, virtual machines existed but were not in the spotlight, with IBM's VM products being one exception. DOS, OS/2, and Windows also had limited DOS virtual machines, though they were more akin to emulation.

1. **VMware**

VMware released a workstation product in 1999, followed by two server products (GSX and ESX) in 2001. This approach allowed guest operating systems to run unmodified.

1. **Denali**

In 2002, the Denali project introduced the concept of paravirtualization. This approach was developed as a workaround for the absence of hardware virtualization support.

1. **x86 Hardware Virtualization Extensions**

In the early 2000s, Intel and AMD introduced hardware support for virtualization to address x86 architecture limitations. This allowed a less privileged execution mode and selective trapping of sensitive instructions.

1. **Hyper-V**

In 2008, Microsoft launched Hyper-V for Windows Server, leveraging x86 hardware virtualization extensions. It provided the option for slower emulation or faster paravirtualization for certain virtual devices, depending on the installation of "Enlightened I/O" extensions in the guest OS.

# **Original Design**

The roots of virtual machines can be traced back to a significant transformation in hardware and software architectures that emerged in the late 1950s. During this era, hardware advancements introduced the concept of multiprogramming, encompassing basic multitasking involving context-switching and basic multiprocessing through dedicated I/O processors and multiple CPUs. The earliest known usage of the term "multiprogramming" can be attributed to Rochester in 1955, describing the IBM 705 system's ability to interrupt an I/O process (such as tape read), execute a process (like calculation) on the data obtained, and then return to the I/O process. The concept of multiprogramming developed further over the decade through work on various systems. Trade-offs concerning security, performance, portability, and complexity, as discussed in the literature on multiprogramming, continue to resonate in contemporary discussions on virtual machines [1].

# **Progress in Development**

Over time, virtual memory has continued to advance, with an emphasis on boosting effectiveness, scalability, and security. Computer memory was very expensive in the 1960s and the early 1970s. With the advent of virtual memory, software systems with high memory requirements could now be operated on machines with less actual memory. All systems were strongly encouraged to transition to virtual memory as a result of the cost reductions.

Then came IBM's System/360 mainframes, which were developed in the 1960s. Dynamic address translation was a technology they created that made it possible to page data into and out of memory as needed. Paging-based virtual memory systems began to proliferate in the 1970s. These systems handled the page allocation to processes and partitioned physical memory into fixed-size blocks (pages). When the Intel 80386 CPU was released in the middle of the 1980s, it came with hardware support for virtual memory, increasing its usability on personal computers.

With the increasing memory requirements of contemporary applications and virtualization technologies, virtual memory has become even more crucial.

# **Current Trends**

* **Machine learning:** Algorithms that can forecast memory access patterns and more effectively distribute resources are utilized to enhance virtual memory management.
* **Non-volatile memory (NVM):** Non-volatile memory (NVM) is increasingly being used in virtual memory systems. To increase performance and save energy, NVM is a sort of memory that keeps its contents even when the power is turned off.
* **Hybrid cloud:** Many businesses are increasingly implementing hybrid cloud architectures, which pair private cloud infrastructure like virtual machines and storage with public cloud services.

# **Importance to OS**

The importance of virtual memory to the operating system is that it enhances the performance, stability, and efficiency of it. It provides Protection and security by preventing processes from accessing and modifying memory areas that they shouldn't, Isolation for Stability, guarantees that each process runs in its own memory space. This will prevent processes from interfering with each other. So, the OS will become more stable, And Efficient in resource management as the virtual memory will allow efficient use of the physical memory. Therefore, the OS can run more processes simultaneously without running out of memory.

# **Challenges**

Page faults occur when a program attempts to access a memory page not in RAM, resulting in a delay as the required data is fetched from slower storage. Effective page replacement algorithms are crucial to minimize the impact of frequent page faults. The complexity of virtual memory, including page table management and replacement policies, introduces overhead and challenges in system design and maintenance. Security concerns arise from vulnerabilities in virtual memory systems, with potential exploits targeting the page table or manipulating the virtual memory to gain unauthorized access. Mitigation strategies, such as robust security measures and regular updates, are essential to address these challenges and ensure the effective operation of virtual memory in modern operating systems.

# **Prospect Future Design Plans**

The development of virtual memory will depend on technological advancements, user demands, and the evolving landscape of computing. The future virtual memory will involve more complicated algorithms for efficient data swapping and better memory allocation strategies. In the context of mobile devices, the future design of virtual memory will optimize memory operation, and this will lead to reduced energy consumption. Virtual memory will need to evolve cloud computing for efficient data sharing and movement between distributed devices. In the future, the user may have control over memory management based on his specific needs and priorities.

# **Program concept**

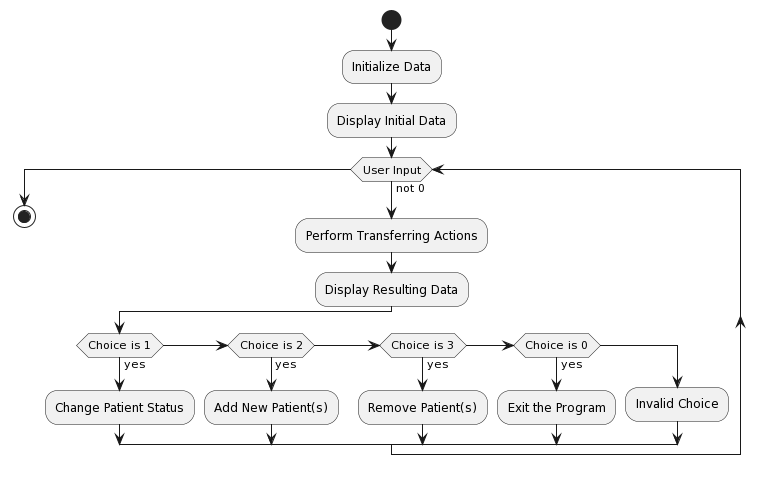
* **Pediatric Surgery Management System**

Pediatrics is the branch of medicine that focuses on the health and medical care of infants, children, and adolescents. Pediatric surgery is a subspecialty within the field of surgery that specifically deals with surgical procedures in infants, children, and adolescents. Sometimes, the child is transferred from general pediatric care to pediatric surgery when he/she has a medical condition or a surgical problem that requires specialized surgical intervention. The timing of the transfer to pediatric surgery depends on the child’s medical condition and needs. In the designed program, the Main memory is represented as the Paediatric surgery, Disk as the Paediatric department, Fixed size frames as beds, and patients/children with their companions as processes. The number of beds required for each patient is determined by the companions. If there is one companion, the number of beds is 2, and if there are two companions, the number of beds is 3. Each child is assigned to a specific status/condition (Critical, Serious, Stable, and Hale) to represent the priority of having surgery. In the valid table (page table), valid children to enter Pediatric surgery are those who have critical or serious conditions only. The program works as follow, the patients with stable condition in the Pediatric surgery are moved back to the Paediatric Department. Then critical and serious cases that exist in the valid table are moved to the Paediatric surgery by the priority (Critical then serious) until there are no more empty beds. Latency has been introduced to simulate the time delay associated with the physical movement of patients between the Pediatric Department and Pediatric Surgery, latency often refers to the time delay between initiating a request and receiving a response. Similarly, in our simulation, latency represents the time delay associated with the transfer of patients between different care units.

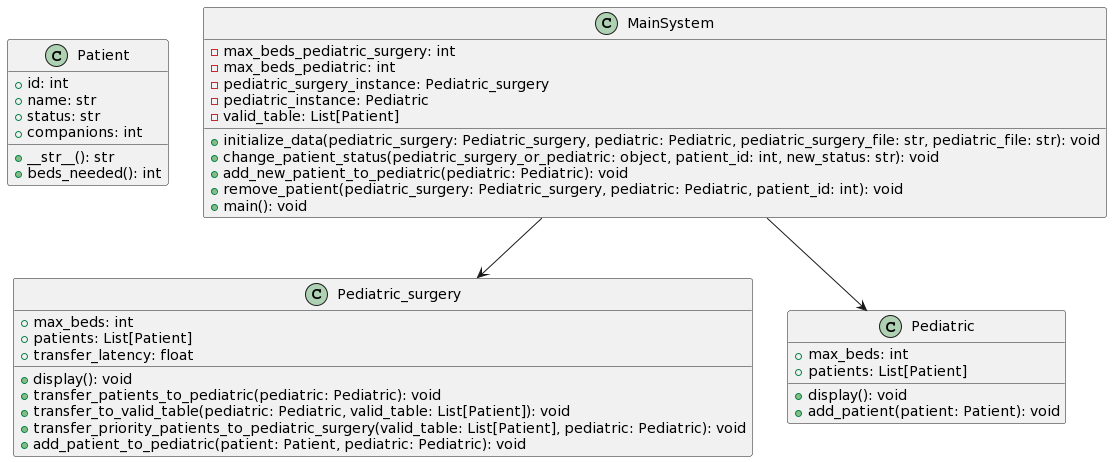
# **Practical part**

1. **Initialization:**
   1. The program starts by creating instances of the `PediatricSurgery` and `Pediatric` classes, representing the main memory and virtual memory, respectively.
   2. Patient data is read from files (`pediatric\_surgery.txt` and `pediatric.txt`) to populate the initial state of the Pediatric Surgery and Pediatric instances.
2. **Initial Display:**
   1. The program displays the initial state of the Pediatric Surgery and Pediatric instances, including information about occupied beds and patient details.
3. **Transfer Actions:**
   1. The program enters a loop to simulate patient transfers:
      1. Patients are transferred from the Pediatric Surgery to the Pediatric based on their health status ('Stable').
      2. Patients from the Pediatric are transferred to a valid table if they have critical or serious health statuses.
      3. Priority patients from the valid table are transferred back to the Pediatric Surgery.
4. **Result Display:**
   1. After each round of simulated transfers, the program displays the resulting state of the Pediatric Surgery and Pediatric instances, including updated information about occupied beds and patient details.
5. **User Interaction:**
   1. The program enters a user interaction loop, where the user can choose from the following options:
      1. **Change Patient Status:**
         1. The user provides the location (Pediatric Surgery or Pediatric), patient ID, and the new health status.
         2. If the status is 'Hale,' the patient is removed; otherwise, the patient's status is updated.
      2. **Add New Patient(s):**
         1. The user can add a new patient to the Pediatric, providing details such as ID, name, status, and number of companions.
      3. **Remove Patient(s):**
         1. The user can remove a patient from either the Pediatric Surgery or Pediatric by providing the patient ID.
      4. **Exit:**
         1. The user can choose to exit the program.
6. **Exit:**
   1. The program exits when the user chooses to do so, terminating the user interaction loop.

Throughout these steps, the program simulates patient transfers, updates patient statuses, adds new patients, removes patients, and provides a user-friendly interface for managing the Pediatric Surgery and Pediatric instances. The cumulative transfer latency is displayed after each round of transfers, reflecting the total time taken for the simulated transfer operations.

**Flow chart of the program:**

**Class Diagram of the program:**



# **References**

[1] Allison Randal. (2020). The Ideal Versus the Real: Revisiting the History of Virtual Machines and Containers. ACM Comput. Surv. 53, 1, Article 5 (January 2021), 31 pages. [The Ideal Versus the Real: Revisiting the History of Virtual Machines and Containers](https://doi.org/10.1145/3365199).