



MEMORY MANAGEMENT

Prepared by Lina Al-fawzan



YEAR 2023 - 1444

Operating System – CIT403
Dr. Fahad Al-Shammari

Abstract

Memory management is the process of controlling and coordinating computer memory, assigning portions called blocks to various running programs to optimize overall system performance and also known as memory allocation. Placement algorithms are implemented to determine the slot that can be allocated process amongst the available ones in the partitioned memory. Memory slots allocated to processes might be too big when using the existing placement algorithms hence losing a lot of space due to internal fragmentation. In dynamic partitioning, external fragmentation occurs when there is a sufficient amount of space in the memory to satisfy the memory request of a process but the process's memory request cannot be satisfied as the memory available is in a non-contiguous manner. This paper describes how to resolve external fragmentation using three allocation algorithms. These algorithms are First-fit, Best-fit and Worst-fit. We will present the implementation of three algorithms and compare their performance on generated virtual trace.

Keywords: Operating system, Memory, Allocation, Segmentation, Best fit, First fit, Worst fit, Performance, Memory Management, Paging.

Contents

Chapter One

1. Introduction.....	4
1.1 Definition.....	4
1.2 Objectives.....	4
1.3 Previous Work.....	4
1.4 Structure.....	5

Chapter Two

2. Memory Management In OS.....	6
2.1 The necessity for memory management.....	6
2.2 Memory management techniques.....	7
2.3 Logical and physical address space.....	8
2.4 Swapping.....	8
2.5 Contiguous memory allocation (Partitioned).....	9
2.6 Contiguous memory allocation (Single).....	12
2.7 Strategies Used for Contiguous Memory Allocation ...	12
2.8 Non-Contiguous Memory Allocation	13
2.9 Paging.....	13
2.10 Segmentation.....	14

Chapter Three

3. Result.....	15
3.1 Methodology.....	15
3.2 Result and Discussion.....	17

Chapter Four

4. Conclusion.....	20
4.1 Future work.....	20
4.2 References.....	20

List of Figures

Fig1:- Uniprogramming memory management(without swapping).....	7
Fig2:- swapping in memory management.....	8
Fig3:- Fixed-size Partition Scheme.....	9
Fig4:- Internal fragmentation problem.....	10
Fig5:- Variable-size Partition Scheme (Dynamic Partitioning).....	11
Fig6:- Noncontiguous allocation.....	13
Fig7:- A flowchart illustrate first fit methodology.....	15
Fig8:- A flowchart illustrate best fit methodology.....	16
Fig9:- A flowchart illustrate worst fit methodology.....	16
Fig10:- Memory allocator simulator GUI.....	17
Fig11.....	17
Fig12	18
Fig13	18
Fig14.....	18

Chapter One

1. Introduction

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. All computer programs, excluding firmware, require an operating system to function.

The main purpose of a computer system to execute programs. These programs with their accusing data must locate in the main memory at the time for execution. The principles of managing the main memory which is one of the most precious resources in a multiprogramming system of an operating system.

Memory consists of a large array of words or bytes each with its own address. Modern operating systems provide efficient memory management and still research is being conducted to improve the way the allocated for applications because the main problem faces the memory allocation algorithm.

1.1 Definition

Memory management is the process of controlling and coordinating a computer's main memory. It ensures that blocks of memory space are properly managed and allocated so the operating system (OS), applications and other running processes have the memory they need to carry out their operations.

As part of this activity, memory management takes into account the capacity limitations of the memory device itself, deallocating memory space when it is no longer needed or extending that space through virtual memory. Memory management strives to optimize memory usage so the CPU can efficiently access the instructions and data it needs to execute the various processes.

Memory management simulation it is a console application which takes the total number of memory blocks, the block sizes, the number of processes, the processes sizes and the allocation method. In a tabular form, the program should show the mapping of the processes to the blocks. It helps computer science students understand the algorithm in a visual way.

1.2 Objectives

To demonstrate the Memory management and implementing a memory allocation simulator by Python.

1.3 Previous Work

The operating system plays a major role in managing the resources. One of the most important resource is the memory. And it is required a scheduling to its input queue to work fairly and efficiently. Many Researchers have introduced various Memory Allocation algorithms from time to time. Some researches that appropriate with our work are: Campus Beaulieu, INSA-IRISA, Rennes, France. (2002) presented a Real-time performance of dynamic memory allocation algorithms. And David A. Barrett, (2013) considered lifetime predictors to improve memory allocation performance Sindhu et al. (2010) proposed an algorithm which can handle all types of process with optimum scheduling criteria.

1.4 Structure

- Chapter 1: an introduction to operating systems and memory management, and the author's goals for what he will achieve in this paper.
- Chapter 2: is about the memory management in operating system. The techniques used in memory management, and the advantages & disadvantages of each of them.
- Chapter 3: a detailed illustration of the memory allocation simulator program that was made by Python.
- Chapter 4: summary the work done in this project and some of the author's personal opinions on what should be done in the future.

Chapter Two

2. Memory Management In OS

The term Memory can be defined as a collection of data in a specific format. It is used to store instructions and process data. The memory comprises a large array or group of words or bytes, each with its own location.

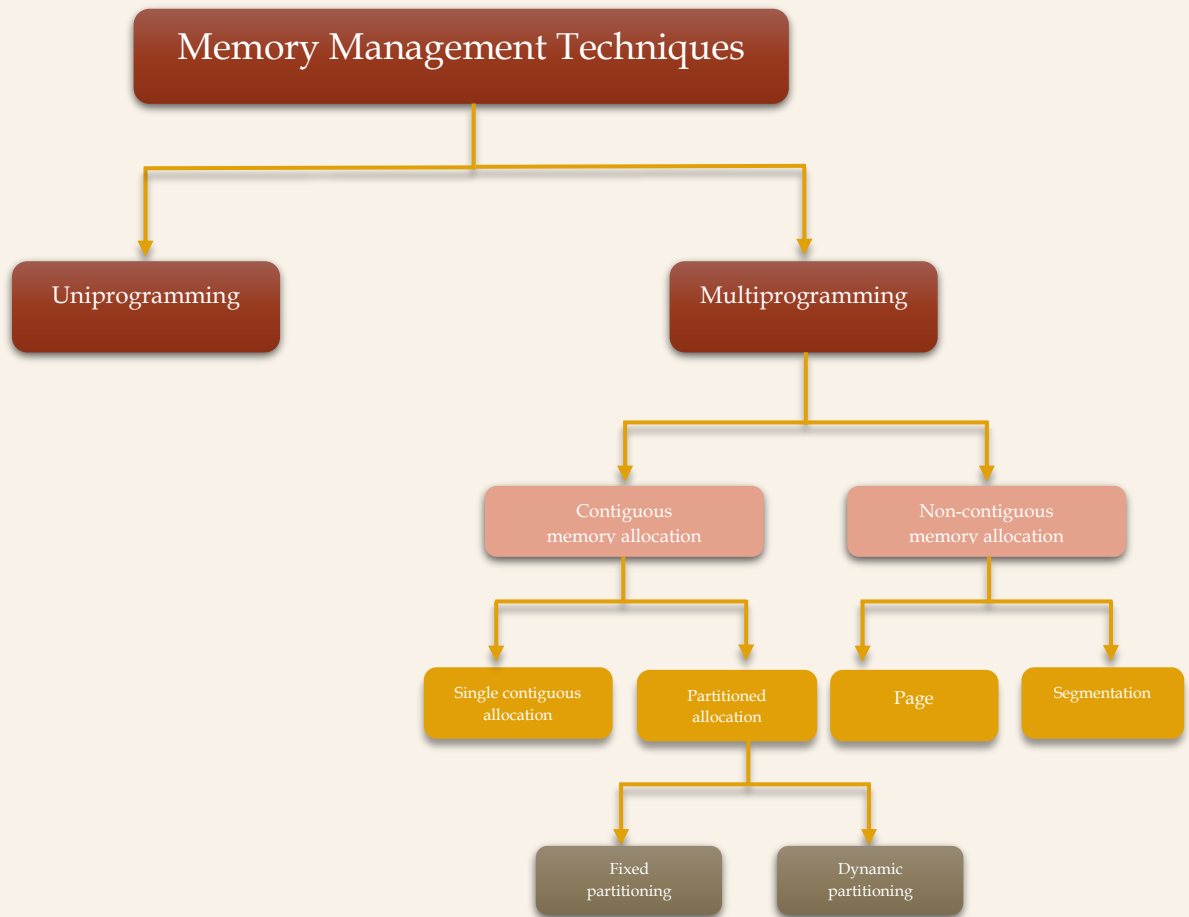
Memory management is **the process of controlling and coordinating a computer's main memory**. It ensures that blocks of memory space are properly managed and allocated so the operating system (OS), applications and other running processes have the memory they need to carry out their operations.

2.1 The necessity for memory management

Why Memory Management is required?

- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

2.2 Memory management techniques



- **Uniprogramming memory management technique:** means **one program** sits in the main memory at a time. Uniprogramming was used in old computers and mobiles. When the computer starts then the operating system and application programs are loaded into main memory. We only count user programs running in RAM. RAM is also called main memory.

In old operating systems (OS) only one program runs on the computer at a time. Either of the browser, calculator or word processor runs at a time. These types of operating systems in which one program runs at a time are known as **Uniprogramming operating systems**.

This is the simplest memory management technique the memory is divided into two sections: one part for operating system, and second part for user program. *Shown in fig1*



Fig1:- Uniprogramming memory management(without swapping).

Advantage:

- It is simple management approach.

Disadvantage:

- It does not support multiprogramming.
 - Memory is wasted.
- **Multiprogramming memory management technique:** operating system may run **many programs** on a single processor computer. If one program must wait for an input/output transfer in a multiprogramming operating system, the other programs are ready to use the CPU. As a result, various jobs may share CPU time. However, the execution of their jobs is not defined to be at the same time period. When a program is being performed, it is known as a "Task", "Process", and "Job". Concurrent program executions improve system resource consumption and throughput as compared to serial and batch processing systems. The primary goal of multiprogramming is to manage the entire system's resources.

2.3 Logical and physical address space

- **Logical Address space:** An address generated by the CPU is known as a "Logical Address". It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.
- **Physical Address space:** An address seen by the memory unit (i.e the one loaded into the memory address register of the memory) is commonly known as a "Physical Address". A Physical address is also known as a Real address. The set of all physical addresses corresponding to these logical addresses is known as Physical address space.

2.4 Swapping

Swapping is a memory management technique **used in multi-programming** to increase the number of processes sharing the CPU. It is a technique of removing a process from the main memory and storing it into secondary memory, and then bringing it back into the main memory for continued execution. This action of moving a process out from main memory to secondary memory is called **Swap Out** and the action of moving a process out from secondary memory to main memory is called **Swap In**. *shown in fig2*

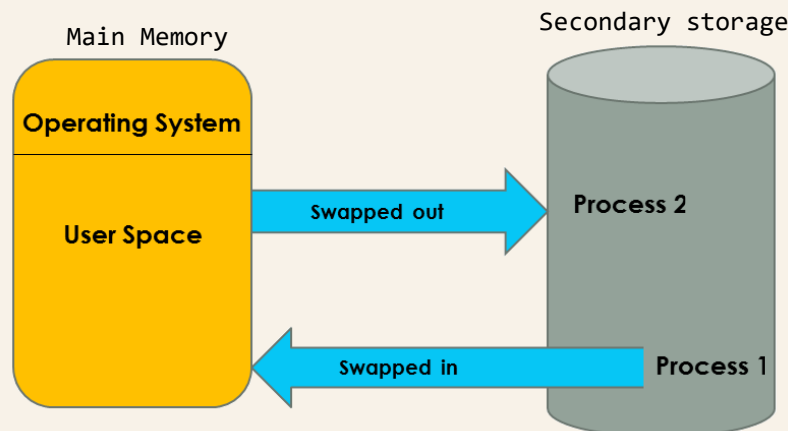


Fig2:- swapping in memory management.

2.5 Contiguous memory allocation (Partitioned)

When a program or process is to be executed, it needs some space in the memory, for this reason some part of the memory has to be allotted to a process according to its requirements. This process is called **memory allocation**.

One such memory allocation technique is **contiguous memory allocation**. As the name implies, we allocate contiguous blocks of memory to each process using this technique. So, whenever a process wants to enter the main memory, we allocate a continuous segment from the totally empty space to the process based on its size.

Whenever a process has to be allocated space in the memory, following the contiguous memory allocation technique, we have to allot the process a continuous empty block of space to reside. This allocation can be done in two ways:

- Fixed-size Partition Scheme
- Variable-size Partition Scheme

Fixed-size Partition Scheme: in this type of contiguous memory allocation technique, each process is allotted a fixed size continuous block in the main memory. That means there will be continuous blocks of fixed size into which the complete memory will be divided, and each time a process comes in, it will be allotted one of the free blocks. Because irrespective of the size of the process, each is allotted a block of the same size memory space. This technique is also called **static partitioning**.

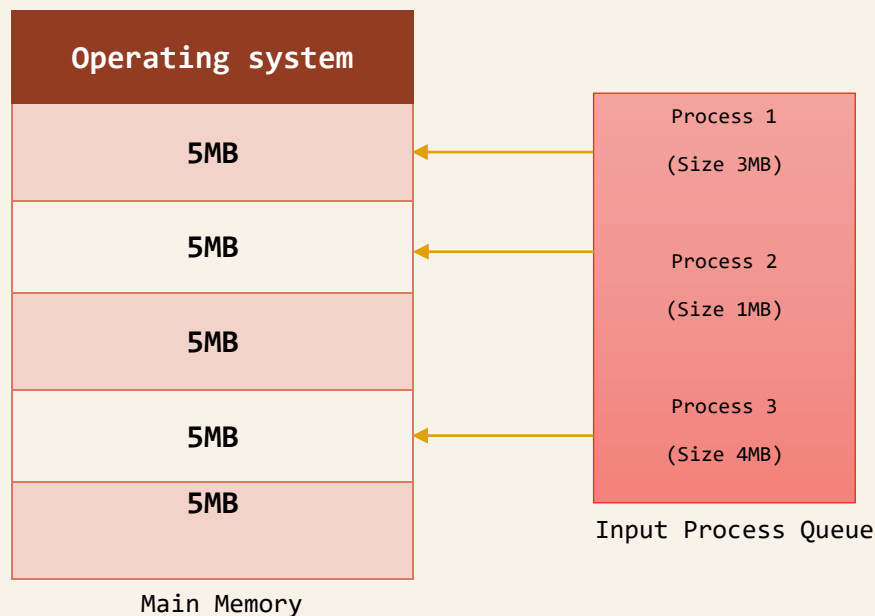


Fig3:- Fixed-size Partition Scheme.

In the diagram above *fig3*, we have 3 processes in the input queue that have to be allotted space in the memory. As we are following the fixed size partition technique, the memory has fixed-sized blocks. The first process, which is of size 3MB is also allotted a 5MB block, and the second process, which is of size 1MB, is also allotted a 5MB block, and the 4MB process is also allotted a 5MB block. So, the process size doesn't matter. Each is allotted the same fixed-size memory block.

It is clear that in this scheme, the number of continuous blocks into which the memory will be divided will be decided by the amount of space each block covers, and this, in turn, will dictate how many processes can stay in the main memory at once.

Advantages

The advantages of a fixed-size partition scheme are:

1. Because all of the blocks are the same size, this scheme is simple to implement. All we have to do now is divide the memory into fixed blocks and assign processes to them.
2. It is easy to keep track of how many blocks of memory are left, which in turn decides how many more processes can be given space in the memory.
3. As at a time multiple processes can be kept in the memory, this scheme can be implemented in a system that needs multiprogramming.

Disadvantages

Though the fixed-size partition scheme has many advantages, it also has some disadvantages:

1. As the size of the blocks is fixed, we will not be able to allot space to a process that has a greater size than the block.
2. The size of the blocks decides the degree of multiprogramming, and only that many processes can remain in the memory at once as the number of blocks.
3. If the size of the block is greater than the size of the process, we have no other choice but to assign the process to this block, but this will lead to much empty space left behind in the block. This empty space could've been used to accommodate a different process. This is called **internal fragmentation**. Hence, this technique may lead to space wastage. Illustrated in *fig4*.

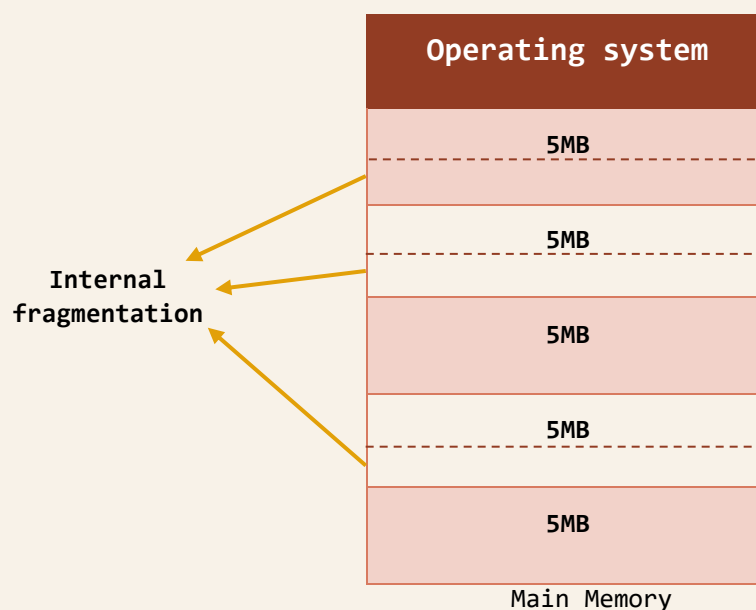


Fig4:- Internal fragmentation problem.

Variable-size Partition Scheme: in this type of contiguous memory allocation technique, no fixed blocks or partitions are made in the memory. Instead, each process is allotted a variable-sized block depending upon its requirements. That means, whenever a new process wants some space in the memory, if available, this amount of space is allotted to it. Hence, the size of each block depends on the size and requirements of the process which occupies it.

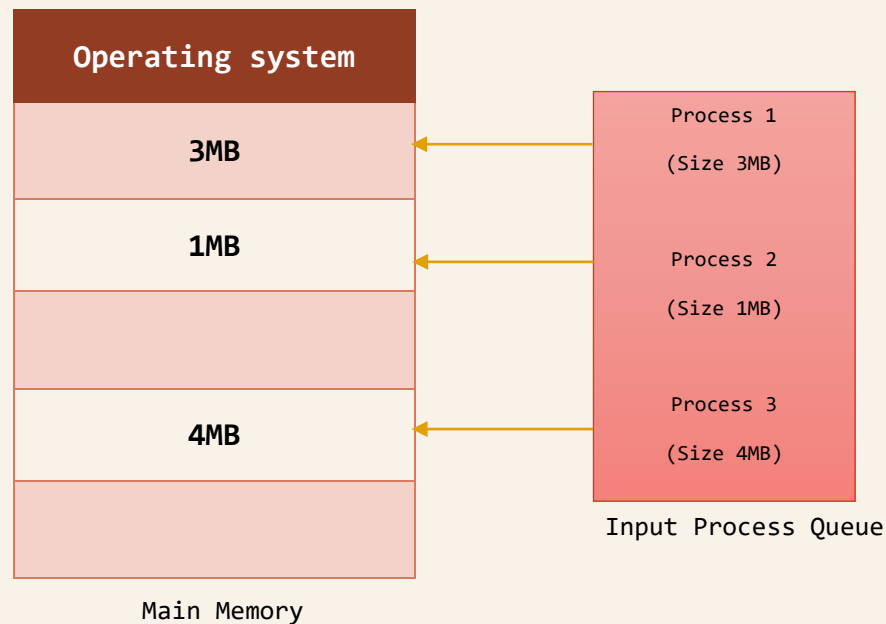


Fig5:- Variable-size Partition Scheme (Dynamic Partitioning).

In the diagram above *fig5*, there are no fixed-size partitions. Instead, the first process needs 3MB memory space and hence is allotted that much only. Similarly, the other 2 processes are allotted only that much space that is required by them.

As the blocks are variable-sized, which is decided as processes arrive, this scheme is also called Dynamic Partitioning.

Advantages

The advantages of a variable-size partition scheme are:

1. As the processes have blocks of space allotted to them as per their requirements, there is no internal fragmentation. Hence, there is no memory wastage in this scheme.
2. The number of processes that can be in the memory at once will depend upon how many processes are in the memory and how much space they occupy. Hence, it will be different for different cases and will be dynamic.
3. As there are no blocks that are of fixed size, even a process of big size can be allotted space.

Disadvantages

Though the variable-size partition scheme has many advantages, it also has some disadvantages:

1. Because this approach is dynamic, a variable-size partition scheme is difficult to implement.
2. It is difficult to keep track of processes and the remaining space in the memory.

2.6 Contiguous memory allocation (Single)

The Single contiguous memory management scheme is the simplest memory management scheme used in the earliest generation of computer systems. In this scheme, the main memory is divided into two contiguous areas or partitions. The operating systems reside permanently in one partition, generally at the lower memory, and the user process is loaded into the other partition.

Advantages of Single contiguous memory management schemes:

- Simple to implement.
- Easy to manage and design.
- In a Single contiguous memory management scheme, once a process is loaded, it is given full processor's time, and no other processor will interrupt it.

Disadvantages of Single contiguous memory management schemes:

- Wastage of memory space due to unused memory as the process is unlikely to use all the available memory space.
- The CPU remains idle, waiting for the disk to load the binary image into the main memory.
- It cannot be executed if the program is too large to fit the entire available main memory space.
- It does not support multiprogramming, i.e., it cannot handle multiple programs simultaneously.

2.7 Strategies Used for Contiguous Memory Allocation

So far, we've seen the two types of schemes for contiguous memory allocation. But **what happens when a new process comes in and has to be allotted a space in the main memory? How is it decided which block or segment it will get?**

Processes that have been assigned continuous blocks of memory will fill the main memory at any given time. However, when a process completes, it leaves behind an empty block known as a **hole**. This space could also be used for a new process. Hence, the main memory consists of processes and holes, and any one of these holes can be allotted to a new incoming process. We have three strategies to allot a hole to an incoming process:

- **First-Fit**

This is a very basic strategy in which we start from the beginning and allot the first hole, which is big enough as per the requirements of the process. The first-fit strategy can also be implemented in a way where we can start our search for the first-fit hole from the place we left off last time.

- **Best-Fit**

This is a greedy strategy that aims to reduce any memory wasted because of internal fragmentation in the case of static partitioning, and hence we allot that hole to the process, which is the smallest hole that fits the requirements of the process. Hence, we need to first sort the holes according to their sizes and pick the best fit for the process without wasting memory.

- **Worst-Fit**

This strategy is the opposite of the Best-Fit strategy. We sort the holes according to their sizes and choose the largest hole to be allotted to the incoming process. The idea behind this allocation is that as the process is allotted a large hole, it will have a lot of space left behind as internal fragmentation. Hence, this will create a hole that will be large enough to accommodate a few other processes.

2.8 Non-Contiguous Memory Allocation

In this type of memory allocation, a process can acquire several memory blocks at different locations in the memory according to its need. The available free space is distributed here and there, unlike contiguous memory allocation, where all the free space is allocated in one place.

The execution of non-contiguous memory is slower than contiguous memory allocation. It **includes segmentation and paging**. There is no loss of memory in this allocation. The processes after swapping can be arranged in any location.

For example, a process having 2 segments, will be allocated in non-contiguous memory blocks in the memory like block 2 (-> P1), block 4 (-> P2).

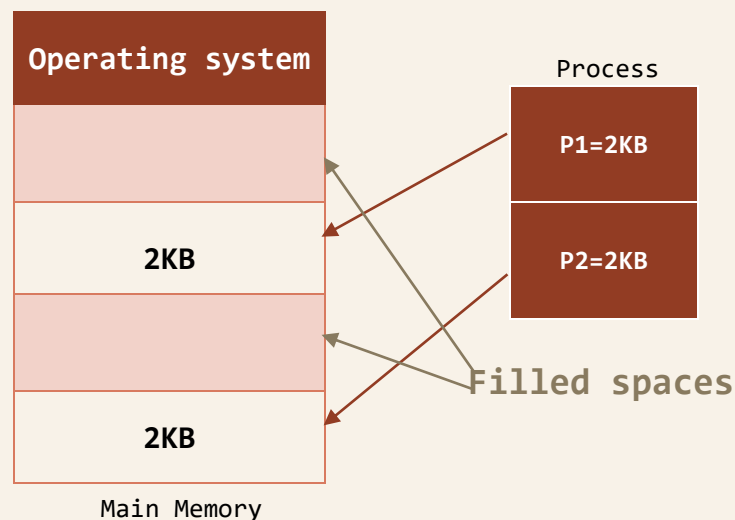


Fig6:- Noncontiguous allocation.

2.9 Paging

Paging is a storage method used in operating systems to recover processes from secondary storage into the main memory in the form of pages. The basic concept behind paging is to break each procedure into individual pages. The primary memory will be separated into frames as well. One page of the process should be saved in one of the memory frames. The pages can be placed in various locations across the memory, but finding contiguous frames or holes is always the goal. Process pages are only brought into the main memory when needed; otherwise, they are stored in secondary storage. Frame sizes vary depending on the operating system. Each frame must have the same size. Because the pages in Paging are mapped to the frames, the page size must be the same as the frame size.

2.10 Segmentation

Segmentation is a memory management method used in operating systems that divides memory into portions of varying sizes. Each component is referred to as a segment, and each segment can be assigned to a process. A table called segment table stores information about each segment. One (or more) segments contain the segment table. The segment table primarily comprises two pieces of information regarding the segment:

Base: This is the segment's starting address.

Limit: The segment's length is the limit.

Chapter Three

3. Result

One of the project objectives is to deliver memory allocation simulations using Python. The program included in this repository simulate the First Fit, Best Fit and Worst Fit memory allocation algorithms used in numerous operating systems. In this chapter, I will illustrate the simulation result.

3.1 Methodology

The simulator allocation program done by two basic steps, the first is understanding the allocation techniques so you can build its algorithm, the second is to develop these algorithms by Python so you can build the allocator simulator program.

Requirements that must be met:

- Programming language: Python3
- Operating system: Windows 11
- Python modules:
 - 1- **OS**: Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.
 - 2- **SYS**: This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.
 - 3- **Win32gui**: Python extensions for Microsoft Windows' Provides access to much of the Win32 API, the ability to create and use COM objects, and the Pythonwin environment.
 - 4- **CustomTkinter**: With CustomTkinter you can create modern looking user interfaces in python with tkinter.

Here, flowcharts of all allocation techniques to illustrate the methodology of each of them:

- Methodology Of First Fit Algorithm:

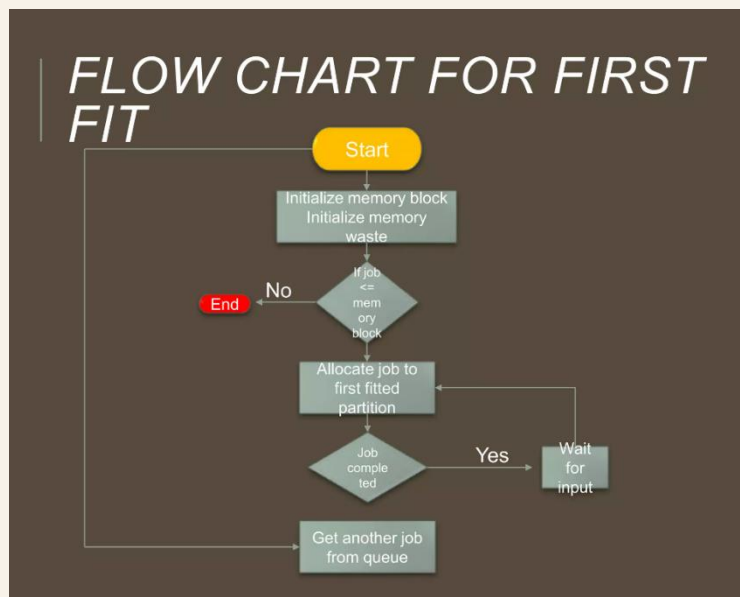


Fig7:- A flowchart illustrate first fit methodology.

- Methodology Of Best Fit Algorithm:

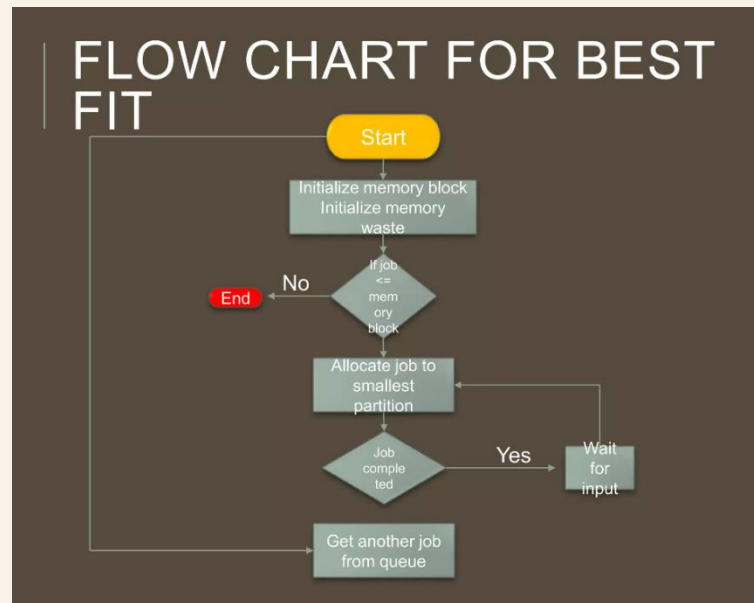


Fig8:- A flowchart illustrate best fit methodology.

- Methodology Of Worst Fit Algorithm:

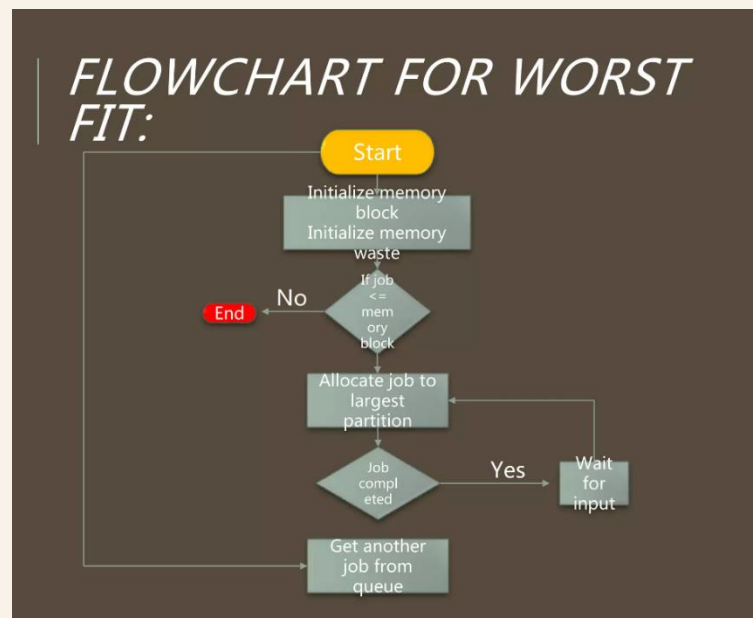


Fig9:- A flowchart illustrate worst fit methodology.

3.2 Result and Discussion

The results are mainly cantered on a graphical interface so that the user can interact with the simulator, and the algorithms of each First fit, Best fit, and Worst fit.

○ GUI

To input the number of partitions and processes that the user wants. As an example, 10 partitions and 3 processes. As shown in *fig8*.

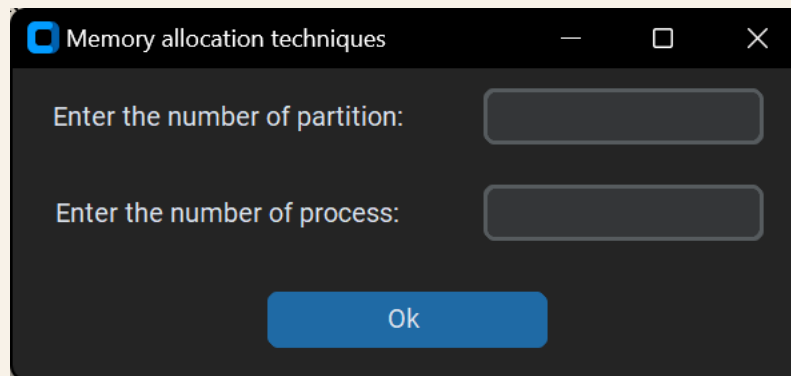


Fig10:- Memory allocator simulator GUI.

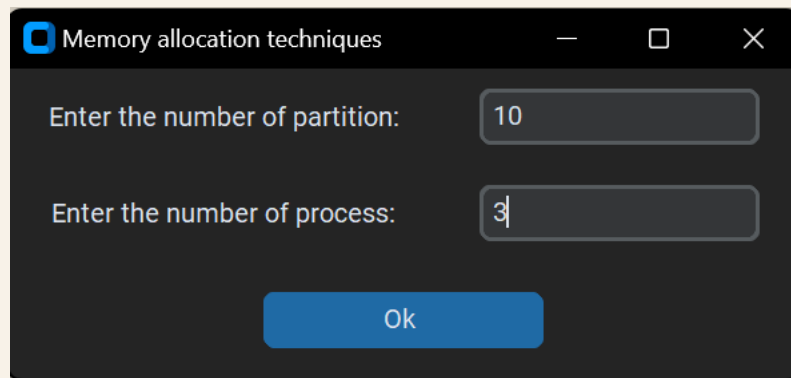
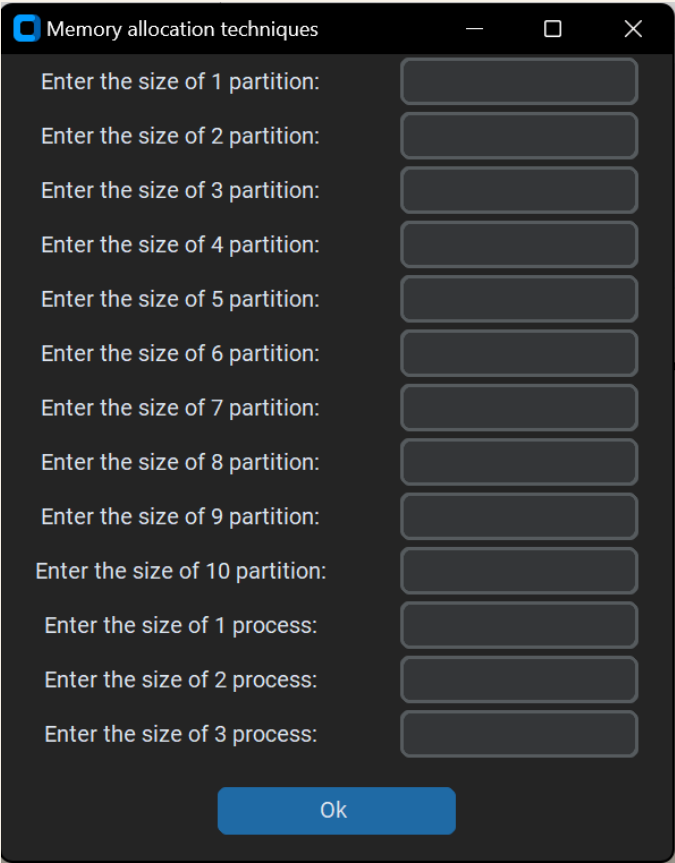


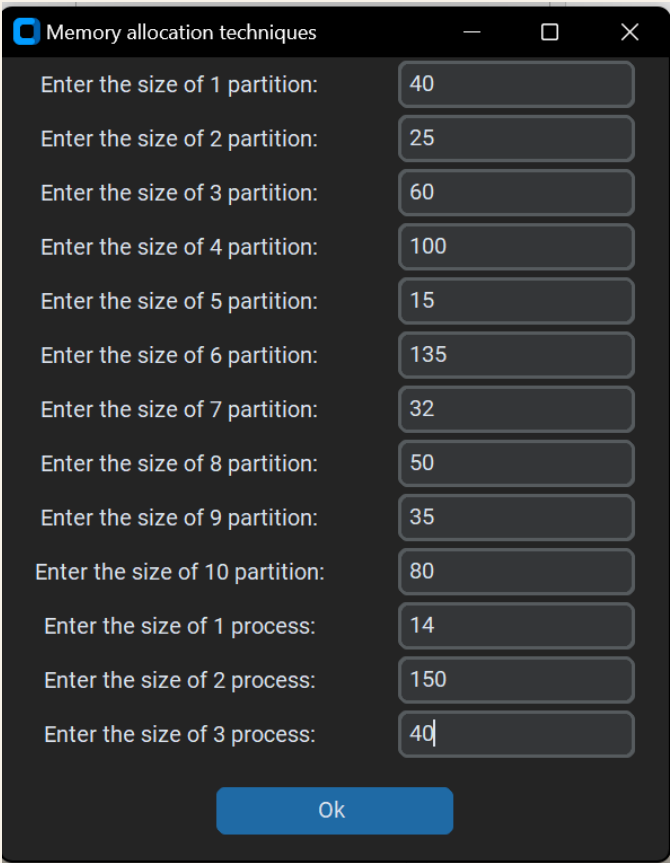
Fig11

Next, the user will be asked about the size of each of the partitions and processes, as shown in *fig9*. For example, I packed them as shown in *fig10*.



A screenshot of a software window titled "Memory allocation techniques". It contains 15 input fields for entering sizes of partitions and processes. The labels are: "Enter the size of 1 partition:", "Enter the size of 2 partition:", "Enter the size of 3 partition:", "Enter the size of 4 partition:", "Enter the size of 5 partition:", "Enter the size of 6 partition:", "Enter the size of 7 partition:", "Enter the size of 8 partition:", "Enter the size of 9 partition:", "Enter the size of 10 partition:", "Enter the size of 1 process:", "Enter the size of 2 process:", and "Enter the size of 3 process:". All input fields are currently empty. At the bottom right is a blue "Ok" button.

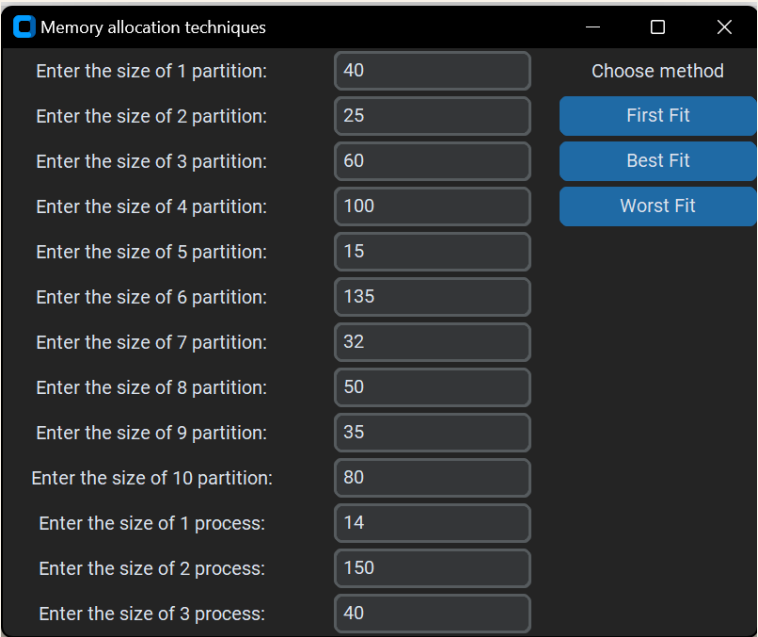
Fig12



A screenshot of the same "Memory allocation techniques" window, but now with numerical values entered in all input fields. The values are: 40, 25, 60, 100, 15, 135, 32, 50, 35, 80, 14, 150, and 40. The "Ok" button remains at the bottom right.

Fig13

After pressing the OK button, the user will be asked for the desired booking method, as shown in *fig11*.



A screenshot of the "Memory allocation techniques" window, showing the same numerical values as Fig13. To the right of the input fields, there is a section titled "Choose method" with three blue buttons: "First Fit", "Best Fit", and "Worst Fit".

Fig14

○ First fit

Memory allocation techniques

Enter the size of 1 partition: 40

Enter the size of 2 partition: 25

Enter the size of 3 partition: 60

Enter the size of 4 partition: 100

Enter the size of 1 process: 14

Enter the size of 2 process: 150

Enter the size of 3 process: 40

Choose method

First Fit

Best Fit

Worst Fit

First Fit Output

Process No. 1 of size 14 allocated to block no. 1

Process No. 2 of size 150 allocated to block no. Not Allocated

Process No. 3 of size 40 allocated to block no. 3

OK

*In the first fit, the partition is allocated which is **first free** sufficient from the top of Main Memory.*

○ Best fit

Memory allocation techniques

Enter the size of 1 partition: 40

Enter the size of 2 partition: 25

Enter the size of 3 partition: 60

Enter the size of 4 partition: 100

Enter the size of 1 process: 14

Enter the size of 2 process: 150

Enter the size of 3 process: 40

Choose method

First Fit

Best Fit

Worst Fit

Best Fit Output

Process No. 1 of size 14 allocated to block no. 5

Process No. 2 of size 150 allocated to block no. Not Allocated

Process No. 3 of size 40 allocated to block no. 1

OK

Done

*In the best fit, the partition is allocating the **smallest free** partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate.*

○ Worst fit

Memory allocation techniques

Enter the size of 1 partition: 40

Enter the size of 2 partition: 25

Enter the size of 3 partition: 60

Enter the size of 4 partition: 100

Enter the size of 1 process: 14

Enter the size of 2 process: 150

Enter the size of 3 process: 40

Choose method

First Fit

Best Fit

Worst Fit

Worst Fit Output

Process No. 1 of size 14 allocated to block no. 6

Process No. 2 of size 150 allocated to block no. Not Allocated

Process No. 3 of size 40 allocated to block no. 6

OK

Done

*In the worst fit, partition is allocated with a process to the which is **largest** sufficient among the freely available partitions available in the main memory.*

Chapter Four

4. Conclusion

Hence, with this we have understood about the indispensable concept of an operating system and its memory management and its segmentation in memory and also a short outlook of memory management. In this paper different memory allocation techniques have been discussed along with their comparative analysis.

Also, a simulate program used for simulating common memory allocation strategies: First-Fit, Best-Fit, and Worst-Fit.

4.1 Future work

This paper presents an algorithm to simulate the memory allocation behavior in contiguous schema, future work includes adding a memory allocation simulator in noncontiguous schema.

4.2 References

[1] GeeksforGeeks. (2023, January 9). *Memory Management in Operating System*.
<https://www.geeksforgeeks.org/memory-management-in-operating-system/>

[2] *Memory Management - javatpoint*. (2011-2021). [www.javatpoint.com](https://www.javatpoint.com/memory-management-operating-system).
<https://www.javatpoint.com/memory-management-operating-system>

[3] Agarwal, S. (2020, May 29). *Uniprogramming and multiprogramming with their differences*
| *Engineer's Portal*. *Engineer's Portal* | Yuvayana.
<https://er.yuvayana.org/uniprogramming-and-multiprogramming-with-their-differences/>

[4] *Contiguous and Non-Contiguous Memory Allocation in Operating System - javatpoint*. (2011-2021).
<https://www.javatpoint.com/contiguous-and-non-contiguous-memory-allocation-in-operating-system>