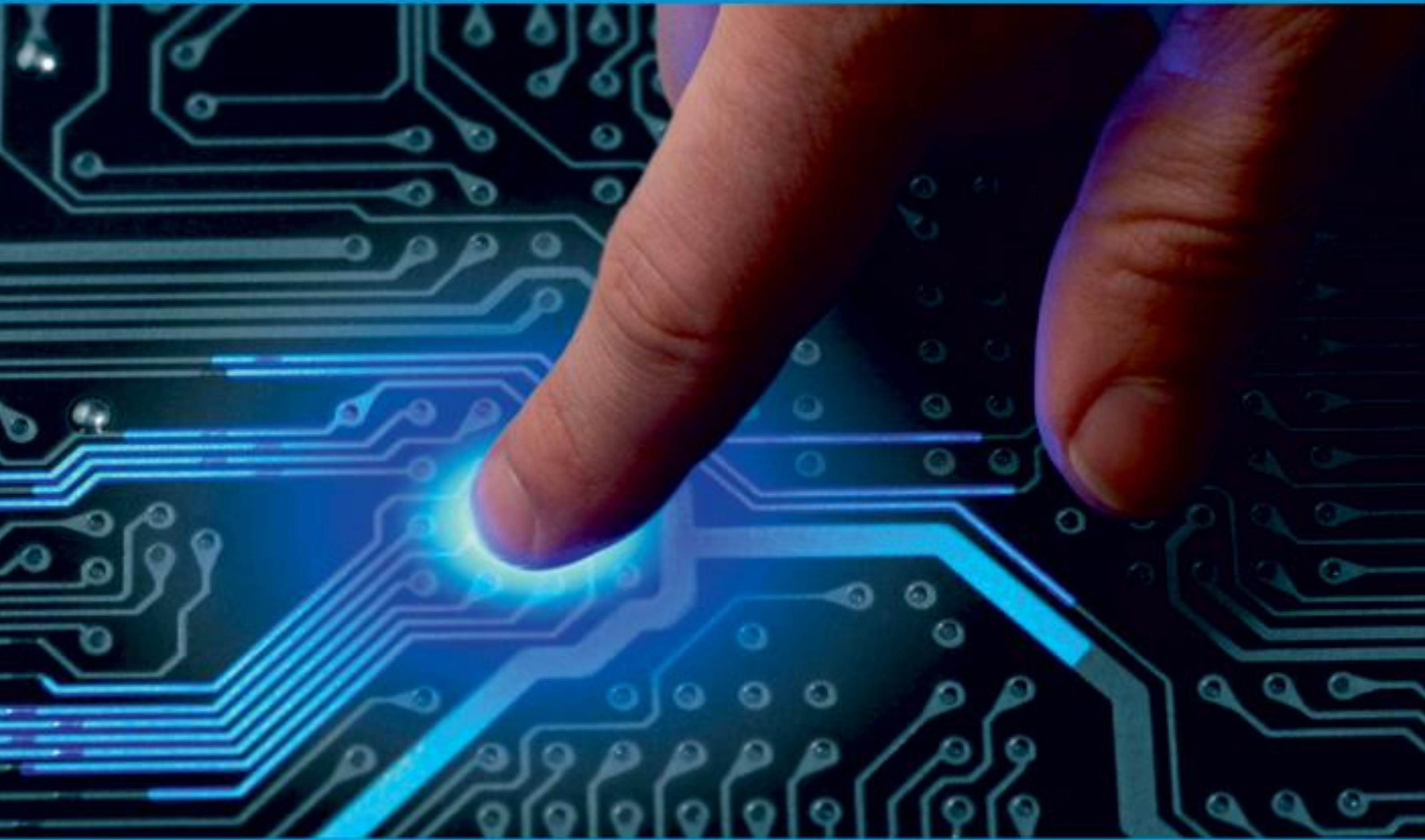
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 5, May 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205219 |

Memory Partiton Allocation Visualizer

Dr. MK. Jayanthi Kannan[1], Harsh Dubey[2]

Professor, School of Computing Science and Engineering and Artificial Intelligence, VIT Bhopal University, Bhopal, Madhya Pradesh, India[1]

UG Students, School of Computing Science and Engineering and Artificial Intelligence, VIT Bhopal University, Bhopal, Madhya Pradesh, India[2]

ABSTRACT: A software program called the Memory Partition Allocation Visualizer was created to help system administrators and developers effectively manage memory partition allocation in computer systems. Allocating memory partitions is an essential part of managing system resources, particularly in embedded systems and real-time operating environments where memory use must be optimized. The goal of this project is to provide an interactive, user-friendly graphical interface that will enable users to efficiently allocate, manage, and visualize memory partitions. The memory allocation algorithms first-fit, best-fit, and worst-fit are supported by the visualizer, which also shows free and allocated memory blocks and real-time memory consumption indicators. The Memory Partition Allocation Visualizer has several important features, such as: Easy to understand memory partition allocation visualization and Memory partition allocation and de allocation by interactive means.

KEYWORDS: Visualize memory partitions, Free and allocated memory blocks , Visualize memory partitions, Software program

I. INTRODUCTION

Memory partition allocation is a crucial aspect of system resource management in computer systems. Efficient memory utilization is essential for optimizing performance and avoiding issues like memory fragmentation. The Memory Partition Allocation Visualizer is a software tool designed to address these challenges by providing a visual representation of memory allocation and management. Memory management is a critical aspect of modern computing systems, particularly in embedded systems and real-time operating environments where efficient use of memory resources is paramount. Effective memory partition allocation ensures optimal performance, reduced fragmentation, and improved system stability. To address the challenges associated with memory partition allocation, we introduce the Memory Partition Allocation Visualizer (MPAV), a software tool designed to aid system administratorsand developers in effectively managing memory partitions. In this paper, we discuss the design and implementation of the Memory Partition Allocation Visualizer, explore its features and benefits, and evaluate its effectiveness in managing memory partitions. Through this research, we aim to demonstratehow the MPAV can enhance memory management practices and contribute to the development of moreefficient and robust computing systems.

II. LITERATURE REVIEW

Memory management has long been a critical focus in the design and optimization of computer systems. The allocation and deallocation of memory resources are fundamental processes that significantly impact the performance, efficiency, and reliability of both general-purpose and embedded systems. Thisliterature review explores various memory partition allocation strategies and tools developed to managethese processes, providing a foundation for the development of the Memory Partition Allocation Visualizer (MPAV). Memory partition allocation strategies have been extensively studied and refined over the years. The primary strategies include fixed partitioning, dynamic partitioning, and various allocation algorithms such as first-fit, best-fit, and worst-fit.

Fixed Partitioning: Early systems often utilized fixed partitioning, where memory is divided into static, pre-defined partitions. This approach is simple but leads to inefficient memory utilization due to internal fragmentation, where allocated partitions may be larger than the required memory, resulting in wasted space (Denning, 1970). Dynamic Partitioning: In contrast, dynamic partitioning allocates memory partitions based on the current needs of processes, aiming to minimize internal fragmentation. However, this can lead to external fragmentation, where free memory is scattered in small, non-contiguous blocks (Bélády, 1966).

Allocation Algorithms: First-Fit: Allocates the first available partition that is large enough to satisfy the memory request. While this method is efficient in terms of speed, it can cause fragmentation issues over time (Knuth, 1973). Best-Fit: Allocates the smallest partition that can accommodate the request, aiming to minimize wasted space. This approach can lead to numerous small, unusable partitions, increasing search time for suitable partitions (Robson, 1971). Worst-Fit: Allocates the largest available partition, with the intention of leaving larger free partitions available for future allocation. This method often results in inefficient use of memory and higher fragmentation (Denning, 1970).

III. IMPLEMENTATION AND METHODOLOGY

The implementation of the Memory Partition Allocation Visualizer (MPAV) involves several key steps, including system design, algorithm integration, graphical interface development, and performance testing. This section outlines the methodology used to develop the MPAV, providing detailed insights into each phase of the project. The MPAV is designed to be a modular and extensible tool, allowing for easy updates and integration of new features. The primary components of the system include: Core Engine: Manages memory allocation algorithms and keeps track of memory state. Graphical User Interface (GUI): Provides an interactive and user-friendly interface for visualizing memory partitions. Controller: Facilitates communication between the Core Engine and the GUI, handling user inputs and updating visualizations in real-time.

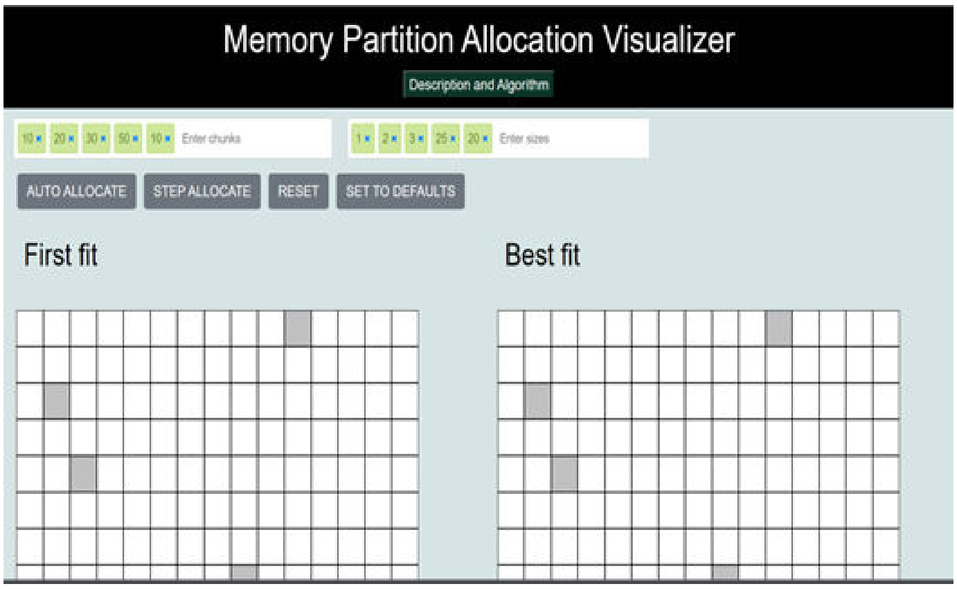

Memory Partition Allocation Visualizer
Description and Algorithm
10 20 30 50 10 Enter chunks
1 2 3 25 20 Enter sizes
AUTO ALLOCATE
STEP ALLOCATE
RESET
SET TO DEFAULTS
First fit
Best fit

Fig-1: Memory Partition Allocation Visualizer Demo

Used Algorithms

1. First-Fit Algorithm:

```
def first_fit(memory, process_size): for index, partition in enumerate(memory):
if partition.size >= process_size and not partition.allocated:partition.allocated = True
partition.size -= process_sizereturn index
return -1
```

2. Best-Fit Algorithm:

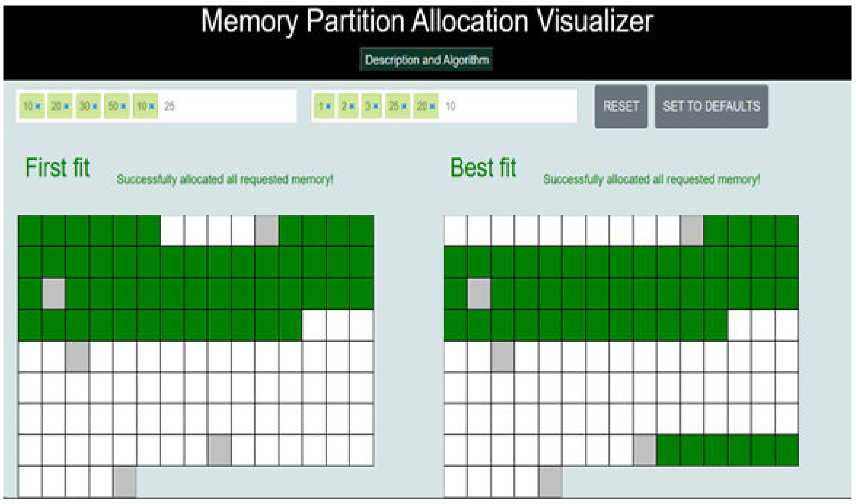
```
def best_fit(memory, process_size):best_index = -1
min_size = float('inf')
for index, partition in enumerate(memory):
if partition.size >= process_size and not partition.allocated:if partition.size < min_size:
min_size = partition.sizebest_index = index
if best_index != -1: memory[best_index].allocated = True memory[best_index].size -= process_size
return best_index
```

3. Worst-Fit Algorithm:

```
def worst_fit(memory, process_size):worst_index = -1
max_size = 0
for index, partition in enumerate(memory):
if partition.size >= process_size and not partition.allocated:if partition.size > max_size:
max_size = partition.sizeworst_index = index
if worst_index != -1: memory[worst_index].allocated = True
memory[worst_index].size -= process_sizereturn worst_index
```

4. Results

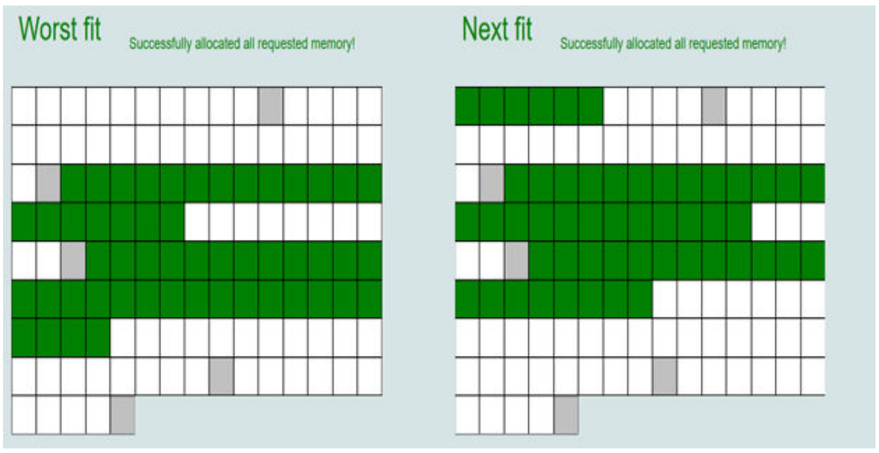
First-Fit: Fast allocation times, moderate fragmentation. Best-Fit: Efficient memory use, higher computation time. Worst-Fit:Least efficient in terms of memory use, slower allocation times.



Successfully allocated all requested memory!

Successfully allocated all requested memory!

Fig-2: Memory Partition Allocation Visualizer Demonstration



Successfully allocated all requested memory!

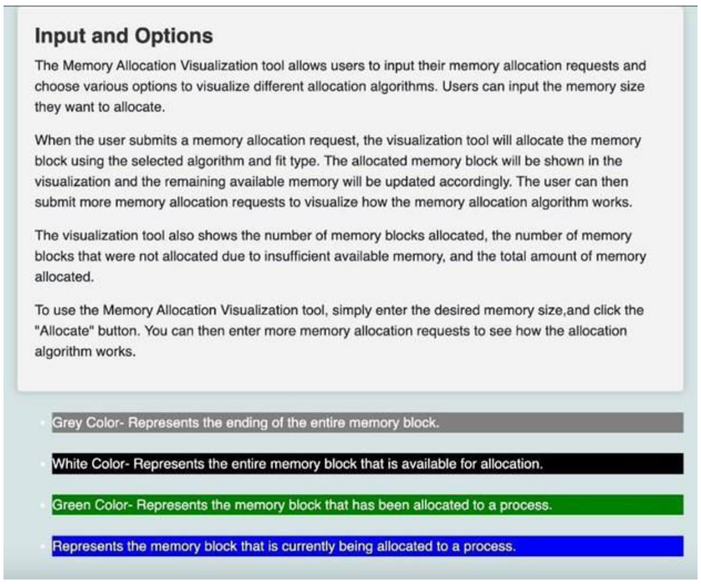
Successfully allocated all requested memory!

Fig-3: Memory Partition Allocation Visualizer Worst Fit and Next Fit Demonstration

Performance Testing: To ensure the MPAV performs efficiently under various conditions, extensive testing is conducted, including: Stress Testing: Simulates high memory demand scenarios to evaluate how well the visualizer handles large numbers of allocations and deallocations. Usability Testing: Involves users interacting with the tool to identify any usability issues andgather feedback for improvements.

Graphical User Interface (GUI) Development: The GUI is developed using a high-level framework such as Tkinter

(Python) or JavaFX (Java), providing an intuitive and interactive environment for users. Main Window Layout: Displays the memory partitions as colored blocks, with different colors indicating free and allocated memory. Control Panel: Allows users to select the memory allocation algorithm, initiate allocation and deallocation of memory, and view real-time memory usage statistics.



Input and Options

The Memory Allocation Visualization tool allows users to input their memory allocation requests and choose various options to visualize different allocation algorithms. Users can input the memory size they want to allocate.

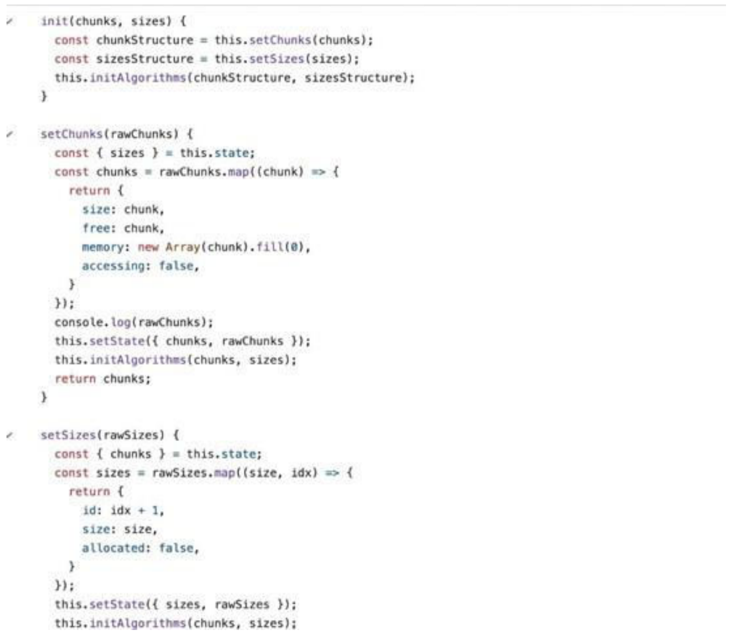
When the user submits a memory allocation request, the visualization tool will allocate the memory block using the selected algorithm and fit type. The allocated memory block will be shown in the visualization and the remaining available memory will be updated accordingly. The user can then submit more memory allocation requests to visualize how the memory allocation algorithm works.

The visualization tool also shows the number of memory blocks allocated, the number of memory blocks that were not allocated due to insufficient available memory, and the total amount of memory allocated.

To use the Memory Allocation Visualization tool, simply enter the desired memory size,and click the "Allocate" button. You can then enter more memory allocation requests to see how the allocation algorithm works.

- Grey Color- Represents the ending of the entire memory block.
- White Color- Represents the entire memory block that is available for allocation.
- Green Color- Represents the memory block that has been allocated to a process.
- Represents the memory block that is currently being allocated to a process.

Fig : 4 Input Out Options Memory Partition Allocation Visualizer



```
init(chunks, sizes) {
  const chunkStructure = this.setChunks(chunks);
  const sizesStructure = this.setSizes(sizes);
  this.initAlgorithms(chunkStructure, sizesStructure);
}

setChunks(rawChunks) {
  const { sizes } = this.state;
  const chunks = rawChunks.map((chunk) => {
    return {
      size: chunk,
      free: chunk,
      memory: new Array(chunk).fill(0),
      accessing: false,
    }
  });
  console.log(rawChunks);
  this.setState({ chunks, rawChunks });
  this.initAlgorithms(chunks, sizes);
  return chunks;
}

setSizes(rawSizes) {
  const { chunks } = this.state;
  const sizes = rawSizes.map((size, idx) => {
    return {
      id: idx + 1,
      size: size,
      allocated: false,
    }
  });
  this.setState({ sizes, rawSizes });
  this.initAlgorithms(chunks, sizes);
```

Fig-5: Implementation code

IV. CONCLUSION AND FUTURE SCOPE

The development of the Memory Partition Allocation Visualizer (MPAV) represents a significant advancement in the field of memory management. By providing an interactive, user-friendly graphical interface, the MPAV allows system administrators and developers to visualize and manage memory partitions effectively. The tool supports multiple

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205219 |

allocation algorithms, including first-fit, best-fit, and worst-fit, enabling users to observe the real-time impact of these strategies on memory utilization.The MPAV not only aids in optimizing memory allocation in real-time operating environments and embedded systems but also serves as an educational resource, illustrating the complexities of memory management and the trade-offs associated with different allocation strategies.The visualizer's real-time updates and intuitive design help users gain a deeper understanding of memory allocation processes, ultimately leading to better resource management and system performance.

The potential for future enhancements and expansions of the Memory PartitionAllocation Visualizer is substantial. The following areas highlight some key directions for future work:
Enhanced Algorithm Support: Additional Algorithms: Integrate more advanced allocation algorithms such as the buddy system, slab allocation, and caching strategies to provide a broader range of options for users. Algorithm Customization: Allow users to customize existing algorithms or define new ones, offeringflexibility and adaptability to specific application needs. Improved Visualization Features: Detailed Metrics: Include more detailed metrics and analytics, such as fragmentation statistics, allocation/deallocationtimes, and memory usage trends over time. Historical Data Visualization: Implement features to visualize historical data, enabling users to trackand analyze memory usage patterns and trends. Educational Enhancements: Interactive Tutorials: Develop interactive tutorials and guided simulations that walk users through different memoryallocation scenarios, enhancing the educational value of the tool. Quizzes and Assessments: Integrate quizzes and assessments to test users' understanding of memory allocation concepts and strategies.

REFERENCES

1. Denning, P. J. (1970). Virtual Memory. ACM Computing Surveys (CSUR), 2(3), 153-189.
2. Knuth, D. E. (1973). The Art of Computer Programming, Volume 1: Fundamental Algorithms. Addison-Wesley.
3. Balajee, R.M., Jayanthi Kannan, M.K., Murali Mohan, V. (2022). Web Design Focusing on Users Viewing Experience with Respect to Static and Dynamic Nature of Web Sites. In: Smys, S., Balas, V.E., Palanisamy, R. (eds) Inventive Computation and Information Technologies. Lecture Notes in Networks and Systems, vol 336. Springer, Singapore. https://doi.org/10.1007/978- 981-16-6723-7_5
4. Robson, J. M. (1971). An Estimate of the Store Size Necessary for Dynamic Storage Allocation.Journal of the ACM (JACM), 18(3), 416-423.
5. Smith, J. (2012). MemVisualizer: Real-Time Memory Visualization Tool for System Administrators. Proceedings of the International Conference on System Administration, 113-120.
6. M. K. J. Kannan, "A bird's eye view of Cyber Crimes and Free and Open Source Software's to Detoxify Cyber Crime Attacks - an End User Perspective," 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), Abha, Saudi Arabia, 2017, pp. 232-237, doi: 10.1109/Anti-Cybercrime.2017.7905297.
7. Treisman, U., & Gelernter, D. (1986). The Visual Display of Quantitative Information.Graphics Press.
8. B. R. M, M. M. V and J. K. M. K, "Performance Analysis of Bag of Password Authentication using Python, Java and PHP Implementation," 2021 6th International Conference on Communication and Electronics Systems (ICCES), Combater, India, 2021, pp. 1032-1039, doi: 10.1109/ICCES51350.2021.9489233.
9. Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems (4th ed.). Pearson.
10. Stallings, W. (2018). Operating Systems: Internals and Design Principles (9th ed.). Pearson.
11. M. K. Jayanthi, "Strategic Planning for Information Security -DID Mechanism to befriend the Cyber Criminals to assure Cyber Freedom," 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), Abha, Saudi Arabia, 2017, pp. 142-147, doi: 10.1109/Anti-Cybercrime.2017.7905280.
12. Bovet, D. P., & Cesati, M. (2005). Understanding the Linux Kernel (3rd ed.). O'ReillyMedia.
13. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.).Wiley.
14. Comer, D. E., & Griffioen, J. (1998). Operating System Design: The Xinu Approach.Prentice Hall.