Process Scheduling in Operating Systems and Evolution of Windows (2023)

Article · June 2023

DOI: 10.6084/m9.figshare.23537031.v1

1 author:

Armaan Sidhu
Manipal University Jaipur

Process Scheduling in Operating Systems and Evolution of Windows (2023)

Armaan Sidhu
3rd Year Computer Science Undergraduate
Manipal University Jaipur, Rajasthan, India

***Abstract*— This review paper discusses the concept of process scheduling in operating systems, with a focus on its implementation in the Windows operating system. The article provides an overview of the process scheduling algorithm used in Windows, which is responsible for determining which process should be given access to the CPU at any given time.**

The article also explores the evolution of the Windows operating system regarding process scheduling. The article discusses how Windows has adapted its process scheduling algorithm over the years to improve performance, reduce latency, and better handle multiple processes running concurrently.

Overall, this technical article provides readers with a detailed understanding of process scheduling in operating systems, as well as an insight into the evolution of the Windows operating system regarding this critical aspect of system performance. This information will be useful to students, researchers, and professionals working in the field of operating system design and development.

***Index Terms*— Process scheduling, Operating systems, Windows OS, CPU, Algorithm, Performance, Latency, Concurrency, Evolution, System design**

I. INTRODUCTION

PROCESS SCHEDULING in modern day operating systems is a fundamental mechanism responsible for managing the allocation of CPU resources to different processes running on the system. Process scheduling is critical to ensuring optimal system performance, reducing latency, and improving overall throughput.

This technical paper provides an overview of process scheduling in operating systems, with a specific focus on the implementation of process scheduling in the Windows operating system. We begin with an introduction to the concept of process scheduling, discussing the factors that influence scheduling decisions and the different algorithms used in modern operating systems.

This work was submitted online for publication. Author Armaan Sidhu is currently pursuing Computer Science Engineering in Third Year at Manipal University Jaipur, Jaipur, Rajasthan – 303007, India (e-mail: armaan.209301656@muj.manipal.edu).

The concept of process scheduling involves deciding which process gets to use the CPU and for how long. The scheduling algorithm should consider various factors such as process priority, CPU utilization, and waiting time. There are various scheduling algorithms used in modern operating systems, including First-Come-First Serve (FCFS), Round Robin (RR), Priority-based, and Shortest Job First (SJF). Each algorithm has its own strengths and weaknesses, making it suitable for different use cases.

The Windows operating system, developed by Microsoft, has evolved significantly since its inception in 1985. One of the key components of the Windows OS is the process scheduler, responsible for managing the allocation of CPU resources to different processes. The Windows process scheduler has undergone significant changes over the years to improve performance, reduce latency, and better handle multiple processes running concurrently.

Fig. 1. *Evolution of Windows 1985 – 2020*. iFunny, https://ifunny.co/picture/evolution-of-windows-windows-1-year-1985-windows-xp-year-voGApLK09, accessed 5th March 2023.

The earliest versions of the Windows operating system used a simple round-robin scheduling algorithm, where each process was given equal access to the CPU in a cyclical manner. However, as the number of processes running on Windows systems increased, this algorithm proved inadequate in handling multiple processes effectively. With the introduction of Windows 95, Microsoft adopted a priority-based scheduling algorithm, where processes with higher priority were given preferential access to the CPU. The introduction of the priority-based algorithm significantly improved the performance of the Windows operating system, reducing latency and improving overall system responsiveness.

Over the years, Microsoft continued to evolve the Windows process scheduler, introducing new algorithms and optimizations to further improve system performance. For example, in Windows Vista, Microsoft introduced a new scheduling algorithm that considered the number of threads associated with a process when making scheduling decisions. This algorithm further improved the performance of the Windows operating system, allowing it to handle increasingly complex workloads.

In recent years, the development of the Windows OS has focused on improving compatibility with new hardware and software, as well as integrating emerging technologies such as virtual reality and artificial intelligence. As such, the evolution of process scheduling in the Windows operating system is likely to continue, with new algorithms and optimizations aimed at improving system performance and meeting the needs of modern users.

A. Abbreviations and Acronyms

The following Abbreviations and Acronyms are related to the contents of this article:

- OS: Operating System
- CPU: Central Processing Unit
- PCB: Process Control Block
- FCFS: First Come-First Serve
- RR: Round Robin
- SJF: Shortest Job First
- MLFQ: Multi-Level Feedback Queue
- I/O: Input/Output
- GUI: Graphical User Interface
- API: Application Programming Interface
- SMP: Symmetric Multi-Processing
- NUMA: Non-Uniform Memory Access
- UMA: Uniform Memory Access
- ACPI: Advanced Configuration and Power Interface
- NUM: Number of User Mode threads
- KLT: Kernel Level Threads
- GVT: Global Virtual Time
- GPU: Graphics Processing Unit
- AI: Artificial Intelligence
- VR: Virtual Reality.

II. Related Works

To give some perspective about the existing and well-known process scheduling states and algorithms, this section throws some light upon this essential discussion for Process Scheduling.

In computer science, process scheduling refers to the technique of assigning computer system resources, particularly CPU time, to different processes that are competing for it. CPU scheduling algorithms determine the order in which processes are executed on a system. The two primary types of process scheduling algorithms are preemptive and non-preemptive.

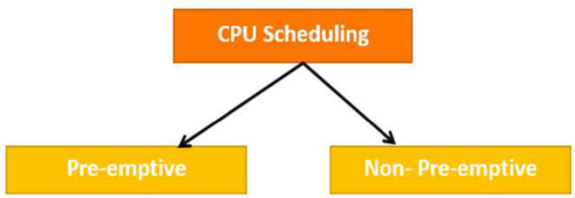

Fig. 2. Classification of *Process Scheduling Algorithms*. Studocu, https://www.studocu.com/in/document/university-of-kashmir/computer-application/os-process-sched-about-os/16391473/, accessed 5th March 2023.

A. Classification of Scheduling Algorithms

- Preemptive Scheduling:

In preemptive scheduling, the CPU can be interrupted and assigned to another process, regardless of whether the current process has completed its execution or not. The CPU scheduler interrupts the currently executing process at regular intervals and assigns the CPU to another process that is waiting in the ready queue.

Preemptive scheduling algorithms require more complex data structures and algorithms than non-preemptive scheduling. They are also known as time-sharing scheduling. Common preemptive scheduling algorithms include Priority Scheduling, Shortest Remaining Job First, Longest Remaining Job First and Round Robin

- Non-preemptive Scheduling:

In non-preemptive scheduling, once a process is assigned to the CPU, it continues to use it until it completes or explicitly releases

the CPU. The CPU is not interrupted until the process either voluntarily relinquishes the CPU or blocks, waiting for an event to occur.

Non-preemptive scheduling algorithms are simple to implement and do not require complex data structures. These algorithms are also known as cooperative scheduling. Common non-preemptive scheduling algorithms include First Come First Served (FCFS), Shortest Job First (SJF), and Priority Scheduling.

In conclusion, the main difference between preemptive and non-preemptive scheduling is that preemptive scheduling allows the CPU to be interrupted and assigned to another process, while non-preemptive scheduling does not. The choice of scheduling algorithm depends on the specific requirements of the system and the nature of the tasks that need to be executed.

B. Process States

Process scheduling in operating systems involves managing the different states that a process can exist in during its lifetime. This Cycle is defined as follows: -

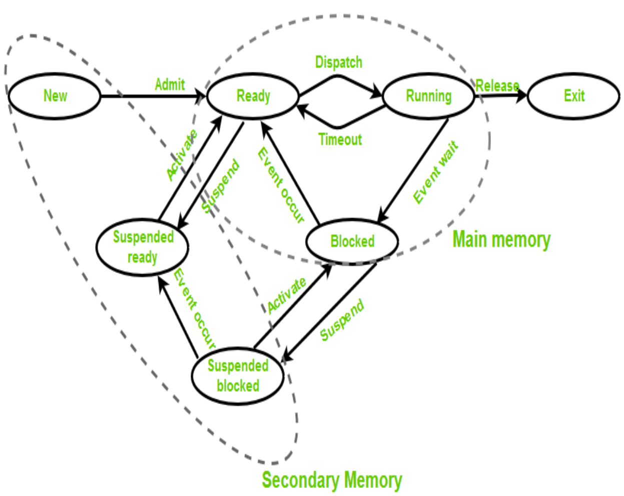

Fig. 3. *States of A Process in Operating System.* GeeksForGeeks. https://www.geeksforgeeks.org/states-of-a-process-in-operating-systems/, accessed March 5th, 2023.

The first stage is "New" which refers to the process being prepared for creation, but not yet created. The operating system retrieves the program from secondary memory to initiate the process.
Once the process is created, it enters the "Ready" state, meaning it is loaded into the primary memory and waiting for the CPU to execute its instructions. A queue maintains a list of all the processes that are ready for CPU execution.

When the CPU selects a process for execution, it enters the "Run" state and the instructions are executed on one of the available CPU cores. If the process requests access to input/output (I/O) devices or other critical regions, it enters the "Blocked" or "Wait" state, during which it waits for completion of the I/O operation without requiring CPU. Once the operation is completed, the process moves back to the "Ready" state.

Once a process is finished, it enters the "Terminated" or "Completed" state, and its PCB (process control block) is deleted. If a process was previously in the "Ready" state but was swapped out of primary memory and onto external storage due to lack of space, it enters the "Suspend Ready" state. It moves back to the "Ready" state when it is again loaded into primary memory.

Similarly, if a process was in the "Blocked" or "Wait" state due to an I/O operation and is swapped out of primary memory, it enters the "Suspend Wait" or "Suspend Blocked" state. It moves back to the "Ready" state once the operation is complete.

C. Classification of Process Schedulers

The scheduling process is typically divided into three types of process schedulers - long-term, medium-term, and short-term schedulers. Each of these schedulers is responsible for different aspects of process management, as explained below:

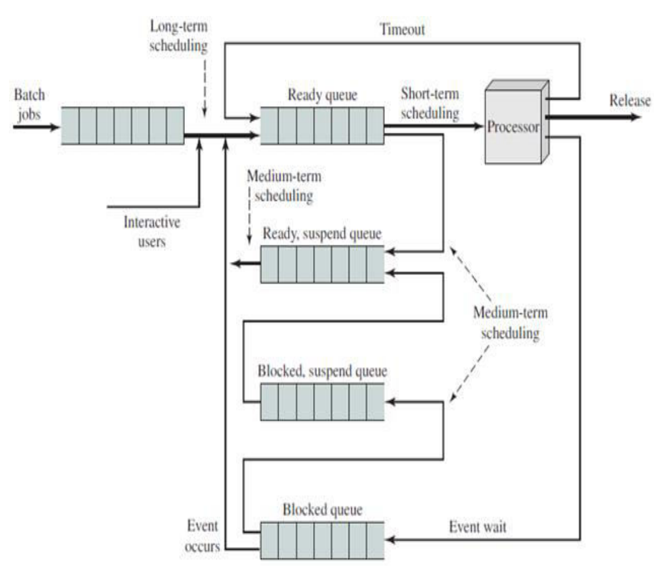

Fig. 4. *Types of Process Schedulers in Operating System.* MyCareerWise. https://mycareerwise.com/content/long-term-short-term-and-mid-term-scheduler/content/exam/nta-

net/computer-science/, accessed March 5th, 2023.

1. Long-Term Scheduler: The Long-Term Scheduler, also known as the Job Scheduler, is responsible for selecting new processes from the pool of available processes and loading them into memory for execution. This scheduler decides which processes are allowed to enter the system and which ones will have to wait. The long-term scheduler controls the degree of multiprogramming, which is the number of processes that can be present in the system memory at any given time. It also considers factors such as CPU utilization, I/O requirements, memory requirements, and priority to determine which processes to select. The primary goal of the long-term scheduler is to balance the system's performance by selecting a set of processes that will maximize the overall system throughput.

2. Medium-Term Scheduler: The Medium-Term Scheduler, also known as the Swapping Scheduler, is responsible for managing the process memory by swapping processes in and out of memory. The medium-term scheduler is not present in all operating systems, but it is commonly used in systems with limited memory resources. When a process is swapped out, its memory contents are saved to disk. When the process is swapped back in, its memory contents are restored from disk. The medium-term scheduler also controls the degree of multiprogramming, but at a higher level than the long-term scheduler. It does this by selecting processes to be swapped out of memory to free up resources for new processes that need to be loaded.

3. Short-Term Scheduler: The Short-Term Scheduler, also known as the CPU Scheduler, is responsible for selecting the process that will execute on the CPU next. It selects from the set of processes that are currently in the memory and ready to execute. The short-term scheduler is responsible for making sure that the CPU is utilized efficiently by selecting the process that will execute for the shortest possible time. It uses algorithms such as round-robin, priority scheduling, or shortest job first to determine which process to select. The primary goal of the short-term scheduler is to ensure that the system is responsive to user requests and that the CPU is always busy executing a process.

In conclusion, the three types of process schedulers work together to manage the system's resources and ensure efficient process execution. The long-term scheduler selects new processes and controls the degree of multiprogramming, the medium-term scheduler manages the process memory by swapping processes in and out of memory, and the short-term scheduler selects the process that will execute on the CPU next. By working together, these schedulers ensure that the system can handle multiple processes simultaneously while providing good performance and responsiveness.

III. SCHEDULING ALGORITHMS

Process scheduling algorithms are an essential component of operating systems, responsible for deciding which process should run next and for how long. These algorithms determine the order in which processes are executed and ensure that the system resources are allocated efficiently and fairly.

There are two main types of process scheduling algorithms: preemptive and non-preemptive. Preemptive algorithms allow the system to interrupt a running process and switch to a higher priority process, while non-preemptive algorithms do not allow interruptions and run the current process to completion before switching to the next process.

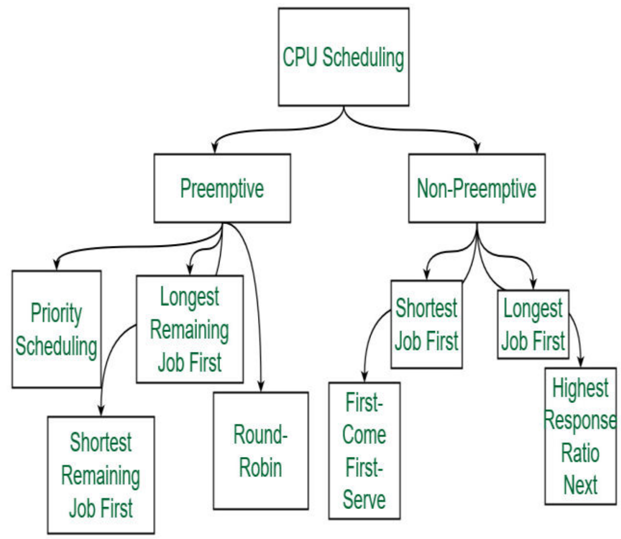

Fig. 2. *Process Scheduling Algorithms*. GeeksForGeeks, https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/, accessed 5th March 2023.

Choosing the right process scheduling algorithm is critical to ensuring optimal system performance, as it affects factors such as response time, throughput, and CPU utilization. Therefore, operating systems use a combination of these algorithms to meet the specific requirements of different applications and user needs.

In this section we explore each Process Scheduling Algorithm In comprehensive detail.

A. *Priority Scheduling*

Priority scheduling is a preemptive process scheduling algorithm in which each process is assigned a priority level based on various criteria, such as the amount of time a process has been waiting, the amount of CPU time it has used, or the relative importance of the process to the system. The process with the highest priority level is executed first.

If two or more processes have the same priority level, the algorithm can use other criteria, such as round-robin scheduling or FCFS, to break the tie.

Priority scheduling is commonly used in real-time systems where some processes are more critical than others and require higher priority levels to ensure they receive the necessary resources. However, one disadvantage of priority scheduling is that lower priority processes may experience starvation, where they may never get the chance to execute if higher priority processes keep arriving. To address this issue, some systems implement aging, where the priority level of a process increases over time, ensuring that every process eventually gets executed.

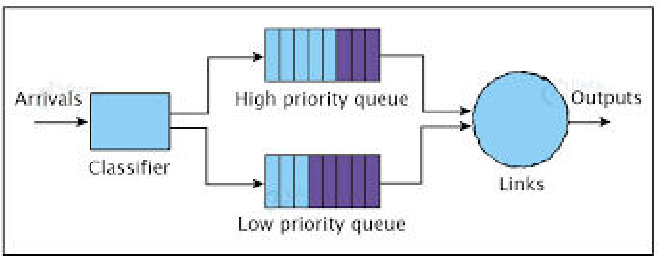

Fig. 5. *Priority Scheduling in Operating Systems*. DataFlair, https://data-flair.training/blogs/priority-scheduling-algorithm-in-operating-system/, accessed on March 5th, 2023.

B. *Longest Remaining Job First (LRJF)*

LRJF is an abbreviation for "Longest Remaining Job First" scheduling algorithm used in computer operating systems. It is a non-preemptive scheduling algorithm that chooses the process with the longest remaining execution time to be executed next.

LRJF is an extension of the SJF (Shortest Job First) algorithm, where instead of choosing the shortest process, we choose the longest. The idea behind this algorithm is to minimize the average waiting time of the processes. The algorithm selects the process with the maximum remaining time, which may lead to the starvation of short processes.

LRJF is a useful scheduling algorithm in batch processing environments where there is no interactive user. It is also used in real-time systems that require long execution times, such as scientific simulations, video rendering, and other intensive computational tasks.

However, LRJF is not suitable for systems with a mix of long and short processes. In such cases, the short processes might starve, causing a higher waiting time, which would be undesirable. Therefore, it is essential to choose the right scheduling algorithm based on the type of workload and the system's requirements.

C. *Shortest Remaining Job First (SRJF)*

SRJF stands for "Shortest Remaining Job First" scheduling algorithm used in computer operating systems. It is a non-preemptive scheduling algorithm that selects the process with the shortest remaining time to be executed next.

SRJF is an extension of the SJF (Shortest Job First) algorithm. In the SJF algorithm, the process with the shortest execution time is executed first, whereas in the SRJF algorithm, the process with the shortest remaining execution time is selected. This means that a process that has already started may be preempted if a new process with a shorter remaining execution time arrives.

SRJF is a useful scheduling algorithm in systems with a mix of long and short processes. It helps in minimizing the average waiting time of the processes. However, it is essential to note that SRJF can lead to starvation of long processes, which may not get a chance to execute until all the short processes have completed.

SRJF is commonly used in interactive systems, where the users' requests are short-lived and need to be processed quickly. It is also used in real-time systems that require fast response times. However, it may not be suitable for batch processing environments where long processes need to be executed.

In conclusion, SRJF is a useful scheduling algorithm that helps in minimizing the average waiting time of processes. However, it is essential to choose the right scheduling algorithm based on the type of workload and the system's requirements.

D. *Round Robin (RR)*

RR (Round Robin) is a scheduling algorithm used in computer operating systems. It is a preemptive scheduling algorithm that assigns a fixed time slice to each process in the ready queue. When the time slice expires, the current process is preempted, and the next process in the queue is selected for execution.

RR is a useful scheduling algorithm in time-sharing systems, where multiple users share a single system simultaneously. It ensures that all the processes get a fair share of the CPU time, preventing any process from monopolizing the CPU. It also helps in improving system responsiveness by giving each process a small time slice to execute, allowing other processes to be executed in the meantime.

RR is a simple and easy-to-implement algorithm that is widely used in operating systems. However, it has some limitations, such as the overhead of context switching, which can affect system performance. Additionally, the choice of the time slice is critical, as a short time slice can cause more context switches, while a long time slice can lead to poor system responsiveness.

Overall, RR is a useful scheduling algorithm that helps in achieving fair CPU time distribution among processes. However, it is essential to choose the right scheduling algorithm based on the system's requirements and workload.

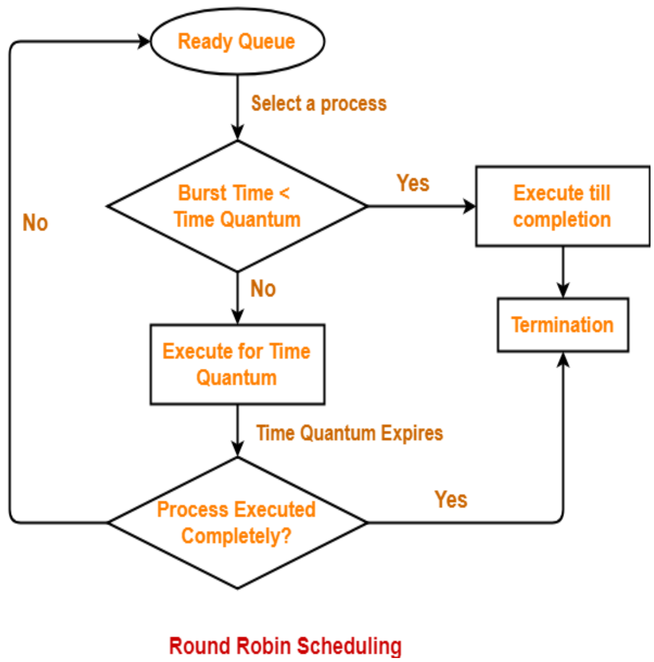

Fig. 5. *Round Robin Scheduling Algorithm*. Gate Vidyalay, https://www.gatevidyalay.com/round-robin-round-robin-scheduling-examples/, accessed on March 5th, 2023.

E. First Come – First Serve (FCFS)

LJF FCFS (First-Come, First-Serve) is a scheduling algorithm used in computer operating systems. It is a non-preemptive scheduling algorithm that selects the process that arrives first to be executed next. It is simple and easy to implement, making it a popular choice for scheduling processes in many systems. However, it has some limitations, such as the potential for long waiting times for processes with longer execution times.

The Algorithm is useful in environments where the processes have similar execution times, and there is no need for prioritization or resource sharing. However, in practice, processes have varying execution times and priorities, which requires more sophisticated scheduling algorithms. FCFS may also suffer from starvation, where some processes may have to wait for a long time before they get a chance to execute, leading to lower system performance and unfairness. This issue can be addressed by using other scheduling algorithms, such as Round Robin or Priority Scheduling, that can handle such uncertainties.

Overall, FCFS is a useful scheduling algorithm that helps in achieving simplicity and fairness in systems with similar execution times and no prioritization or resource sharing requirements. However, it is essential to choose the right scheduling algorithm based on the system's requirements and workload.

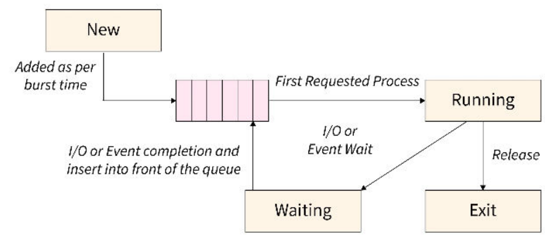

Fig. 5. *FCFS Scheduling Algorithm*. Scaler, https://www.scaler.com/topics/c-program-for-fcfs-scheduling/, accessed on March 5th, 2023.

F. Shortest Job First (SJF)

SJF (Shortest Job First) is a scheduling algorithm used in computer operating systems. It is a non-preemptive scheduling algorithm that selects the process with the shortest execution time to be executed next. It is based on the idea that shorter processes have a lower waiting time and should be executed first. It is useful in batch processing environments, where there is no interactive user, and the system can prioritize the execution of shorter jobs to minimize the average waiting time of the processes.

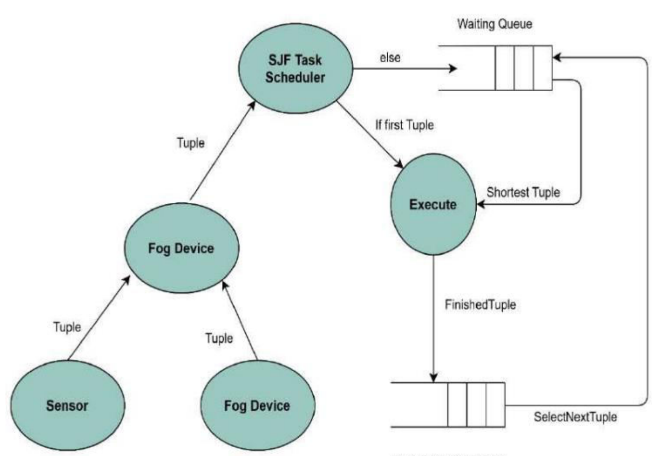

Fig. 5 *SJF Scheduling Algorithm*. Amer Sallam and 8 Others, https://www.researchgate.net/publication/351059407_Perfor

mance_Evaluation_of_Fog-Computing_Based_on_IoT_Healthcare_Application/, accessed on March 5th, 2023.

The Algorithm has some limitations, such as the lack of responsiveness to shorter processes. Longer processes may monopolize the CPU, leading to the starvation of shorter processes, which would have a higher waiting time. This issue can be addressed by using the preemptive version of the algorithm, known as Shortest Remaining Time First (SRTF), which selects the process with the shortest remaining execution time, instead of the shortest execution time. It is a useful scheduling algorithm in systems where the processes have known execution times. However, in practice, it is difficult to estimate the exact execution time of a process. Therefore, it is essential to use other scheduling algorithms, such as Round Robin or Priority Scheduling, that can handle such uncertainties.

In conclusion, SJF is a useful scheduling algorithm that helps in minimizing the average waiting time of processes. However, it has some limitations that need to be addressed in practice. Therefore, it is essential to choose the right scheduling algorithm based on the system's requirements and workload.

G. Longest Job First (LJF)

LJF (Longest Job First) is a scheduling algorithm used in computer operating systems. It is a non-preemptive scheduling algorithm that selects the process with the longest execution time to be executed next. It is based on the idea that longer processes have a higher waiting time and should be executed first. It is useful in batch processing environments, where there is no interactive user, and the system can prioritize the execution of longer jobs to minimize the average waiting time of the processes.

The Algorithm has some limitations, such as the potential for short processes to starve. If the system is overloaded with long processes, short processes may have to wait a long time before they get a chance to execute, leading to higher waiting times. This issue can be addressed by using the preemptive version of the algorithm, known as Longest Remaining Time First (LRTF), which selects the process with the longest remaining execution time, instead of the longest execution time. It is a useful scheduling algorithm in systems with long-running jobs, such as scientific simulations or video rendering. However, it may not be suitable for systems with a mix of long and short processes, where short processes need to be executed quickly.

In conclusion, LJF is a useful scheduling algorithm that helps in minimizing the average waiting time of processes in batch processing environments. However, it has some limitations that need to be addressed in practice. Therefore, it is essential to choose the right scheduling algorithm based on the system's requirements and workload.

H. Highest Response Ratio Next (HRRN)

HRRN (Highest Response Ratio Next) is a scheduling algorithm used in computer operating systems. It is a non-preemptive scheduling algorithm that selects the process with the highest response ratio to be executed next. The response ratio is calculated as the ratio of the waiting time plus the execution time of a process to its execution time. The idea behind HRRN is that processes with a high response ratio have waited longer and deserve to be executed first, providing better performance and fairness than other scheduling algorithms.

The Algorithm is useful in environments where the processes have varying execution times and waiting times. It can adapt to changes in the workload and prioritize processes that have been waiting for a long time, resulting in a lower average waiting time and better system performance. However, it has some limitations, such as the complexity of the calculation of the response ratio, which can affect the system's performance. Additionally, HRRN may suffer from starvation if a process with a low response ratio keeps getting preempted by processes with higher response ratios.

Overall, HRRN is a useful scheduling algorithm that helps in achieving better performance and fairness in systems with varying execution times and waiting times. However, it is essential to choose the right scheduling algorithm based on the system's requirements and workload.

IV. EVOLUTION OF WINDOWS OPERATING SYSTEM

Windows operating system has undergone significant evolution over the years, from its first version, Windows 1, in 1985 to Windows 11 in 2021. The evolution of Windows has been marked by significant changes in terms of technology, design, and functionality. The major events in the Evolution of Windows Operating System are: -

- Windows 1: It was the first version of the Windows operating system, released in 1985. It was a graphical user interface (GUI) operating system that required a lot of memory and processing power to run smoothly. Windows 2, released in 1987, was designed to be a standalone operating system and introduced several new features, including the ability to overlap windows and support for expanded memory.

- Windows 3: It was released in 1990, was a significant improvement over Windows 2. It introduced improved graphics, improved memory management, and support for TrueType fonts. Windows 95, released in 1995, was a major milestone in the evolution of Windows operating system. It introduced several new features, including the taskbar, start menu, and plug-and-play support for hardware. Windows 95 was a commercial success and was widely adopted by businesses and consumers alike.

- Windows 98: It was released in 1998, was an improvement over Windows 95. It introduced support for USB devices, improved hardware support, and improved networking capabilities. Windows 2000, released in 2000, was designed for businesses and enterprise users. It introduced features such as Active Directory, support for symmetric multiprocessing (SMP), and support for the NTFS file system.

- Windows XP: It was released in 2001, was a significant improvement over Windows 2000. It introduced improved security, improved networking, and a new user interface. Windows XP was also the first version of Windows to support Windows Update, which allowed users to download and install security patches and updates automatically.

- Windows Vista: It was released in 2006, was a significant departure from previous versions of Windows. It introduced a new user interface, improved security, and improved networking. However, Windows Vista was criticized for its slow performance and high system requirements. Many users chose to stick with Windows XP instead of upgrading to Windows Vista.

- Windows 8: It was released in 2012 and represented a significant departure from previous versions of Windows. It introduced a new user interface, called Metro, which was designed to work well on touch-enabled devices. However, the Metro interface was not well-received by many users and was criticized for being confusing and difficult to navigate.

- Windows 8.1: It was released in 2013, addressed many of the criticisms of Windows 8 and reintroduced some of the familiar features of previous versions of Windows, such as the Start button. It also introduced new features such as support for 3D printing and improved support for high-resolution displays.

- Windows 10: It was released in 2015, was designed to be a universal operating system that could run on a wide range of devices, from desktops to tablets and smartphones. It introduced several new features, such as Cortana, Microsoft's digital assistant, and a new web browser called Microsoft Edge. Windows 10 also introduced the Windows Insider Program, which allowed users to test new features and provide feedback to Microsoft.

- Windows 11: It was released in 2021, is the latest version of Windows operating system. It introduces a new design language, called Fluent Design, which is designed to be more modern and streamlined. Windows 11 also introduces new features such as Snap Layouts, which allow users to easily arrange and resize multiple windows on their screen, and support for Android apps through the Microsoft Store.

To summarize, the evolution of Windows operating system from Windows Vista in 2006 to Windows 11 in 2021 has been marked by significant changes and improvements in terms of design, functionality, and technology. Each new version of Windows has introduced new features and addressed the issues of its predecessor, with Windows 11 representing the latest and most advanced version of the operating system.

V. COMPARITIVE STUDY OF SCHEDULING ALGORITHMS

This section deals with the concise and precise comparative study of all the Process Scheduling Algorithms used in the Modern day discussed so far.

TABLE I
Comparison of Preemptive Scheduling Algorithms

Scheduling Algorithm	Priority Scheduling	Shortest Remaining Job First	Longest Remaining Job First	Round Robin
Type	Preemptive	Preemptive	Preemptive	Preemptive
Criteria for Scheduling	Priority	Remaining CPU burst time	Remaining CPU burst time	Time Slice
Advantages	High-priority jobs are executed first; suitable for real-time systems	Efficient use of CPU; reduces average waiting time	Suited for long, computation-intensive jobs; avoids starvation of long jobs	Suitable for interactive systems; prevents long jobs from monopolizing the CPU
Disadvantages	Low-priority jobs may suffer from starvation; priority inversion may occur	May lead to starvation of longer jobs; requires knowledge of the total CPU burst time	Shorter jobs may be starved; requires knowledge of the total CPU burst time	Low-priority jobs may suffer from starvation; large time slices may lead to poor performance
Example	Air traffic control systems	Scientific computing	Batch processing	Desktop computing
Implementation Difficulty	Medium	Medium	Low	Low

TABLE II
Comparison of Non - Preemptive Scheduling Algorithms

Scheduling Algorithm	First Come - First Serve	Shortest Job First	Longest Job First	Highest Response Ratio Next
Type	Non-preemptive	Non-preemptive	Non-preemptive	Non-preemptive
Criteria for Scheduling	Arrival time	CPU burst time	CPU burst time	Response ratio
Advantages	Simple and easy to implement; suited for long CPU-bound jobs	Efficient use of CPU; reduces average waiting time	Suited for short, interactive jobs; prevents starvation of shorter jobs	Provides a balance between priority and length of job
Disadvantages	Poor response time for short jobs; may result in long waiting times for later jobs	Requires knowledge of all CPU burst times; may lead to starvation of longer jobs	May result in long waiting times for longer jobs	Requires calculations of response ratios; may result in starvation of longer jobs
Example	Batch processing	Scientific computing	Desktop computing	Interactive computing
Implementation Difficulty	Low	Medium	Medium	Medium

VI. Conclusion

To conclude, process scheduling is an essential part of any operating system that helps in managing the allocation of system resources to running processes. It ensures efficient utilization of CPU, fair distribution of system resources among multiple processes, and a responsive and interactive user experience.

The evolution of Windows operating system's process scheduling since its inception with Windows 1 in 1985 shows a clear focus on improving the efficiency and responsiveness of the system. From the basic first come first serve scheduling algorithm to the advanced Round Robin scheduling algorithm in Windows 11, the operating system has come a long way.

With the addition of preemptive scheduling algorithms like Priority Scheduling and Shortest Remaining Job First, Windows Vista brought significant improvements to the system's ability to handle multiple processes and prioritize critical tasks. The latest version of Windows, Windows 11, continues to prioritize efficient resource allocation and responsive user experience using advanced scheduling algorithms like Round Robin and Preemptive Priority Scheduling.

The evolution of process scheduling algorithms in Windows OS is driven by the changing requirements of modern computing, from desktop computing to interactive and real-time systems. As computing continues to evolve, process scheduling will remain a crucial component of operating systems, ensuring that system resources are allocated efficiently and fairly to running processes.

In summary, the evolution of process scheduling in Windows OS is a testament to the ongoing development and refinement of these algorithms, driven by the need for more efficient and responsive computing systems. The importance of process scheduling in operating systems cannot be overstated, and it will continue to play a crucial role in ensuring the smooth functioning of modern computing systems.

Appendix

Glossary

- Process: A program that is currently executing in the computer system.
- CPU: Central Processing Unit. It is the primary component of a computer system that performs the instructions of a computer program.
- Process scheduling: The mechanism by which an operating system selects a process to run on the CPU.
- Scheduling algorithm: A set of rules that determines the order in which processes are selected for execution on the CPU.
- Preemptive scheduling: A scheduling algorithm that allows a higher priority process to interrupt a lower priority process that is currently executing.
- Non-preemptive scheduling: A scheduling algorithm that allows a process to continue executing until it completes or blocks itself, without interruption from higher priority processes.
- First Come First Serve (FCFS) scheduling: A non-preemptive scheduling algorithm that selects the process that arrives initially in the ready queue.
- Shortest Job First (SJF) scheduling: A non-preemptive scheduling algorithm that selects the process with the shortest expected processing time next.
- Longest Job First (LJF) scheduling: A non-preemptive scheduling algorithm that selects the process with the longest expected processing time next.
- Priority Scheduling: A preemptive scheduling algorithm that selects the process with the highest priority next.
- Round Robin (RR) scheduling: A preemptive scheduling algorithm that assigns a fixed time slice to each process in the ready queue, allowing each process to execute for a set amount of time before being preempted.

- Response time: The time taken by an operating system to start executing a process after it has been submitted.

- Throughput: The number of processes that can be completed in a unit of time.

- Fairness: The extent to which the operating system provides equal opportunity to all processes to access the CPU and system resources.

References and Footnotes

A. References

- Silberschatz, A., Galvin, P.B., & Gagne, G. (2018). Operating System Concepts, 10th edition. John Wiley & Sons, Inc.

- Tanenbaum, A.S., & Bos, H. (2015). Modern Operating Systems, 4th edition. Pearson Education, Inc.

- Stallings, W. (2018). Operating Systems: Internals and Design Principles, 9th edition. Pearson Education, Inc.

- Microsoft Corporation. (2021). Windows 11: Features and System Requirements. Retrieved from https://www.microsoft.com/en-us/windows/windows-11/

- Microsoft Corporation. (2021). A history of Windows. Retrieved from https://www.microsoft.com/en-us/windows/windows-10-os-history/

- IEEE Standard for Information Technology – POSIX Part 1: System Application Program Interface (API) [C Language] (2018). IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008).

B. Footnotes

1. Windows 10 introduced several new features, including the virtual desktops feature and the Cortana voice assistant.

2. In "Shortest Remaining Job First" scheduling, the scheduler selects the process with the shortest remaining burst time for execution.

3. In "Round Robin" scheduling, each process is executed for a fixed time slice or quantum, after which it is preempted and replaced with the next process in the queue.

4. The term "time slice" refers to the amount of time allocated to each process in Round Robin scheduling.

5. Windows Vista introduced several new features, including Aero glass interface and enhanced security features.

6. The priority of a process can be adjusted dynamically during runtime in Preemptive Priority Scheduling.

7. The term "burst time" refers to the amount of time required by a process to complete its execution in CPU Scheduling algorithms.

Acknowledgment

I would like to express my sincere gratitude to all those who have contributed to the completion of this technical article.
First and foremost, I would like to thank my colleagues and mentors for their guidance, insights, and feedback throughout the research and writing process. Their expertise and support have been invaluable in shaping the content and improving the quality of the article.

I would also like to thank the academic and research communities for their continuous efforts in advancing the field of cryptography, and for sharing their knowledge and findings through publications, conferences, and online resources.

Finally, I would like to acknowledge the inspiration and motivation provided by the digital world and its ongoing evolution, which underscores the need for continued research and development in Operating Systems field to ensure the users get an all-round experience.

Once again, thank you to everyone who have contributed to this article in any way, shape, or form.

Armaan Sidhu born 11th October 2003 in Gwalior, Madhya Pradesh, India is an Undergraduate Engineering Student currently pursuing *computer science with cyber security as program elective* in Penultimate Year at Manipal University Jaipur, Rajasthan.

He has gained a great amount of experience by working upon several tasks in Internships, with work experience including (but not limited to):

- Web Development remote internship, started January 2023, at WictroniX, a startup providing B2B IT Services and Consultations to Established Companies, launched in Late 2022 and registered by the Government of India

- Content Writing remote internship, June 2021 – March 2022 at smartwords.in, an online blogging website.

The Author is the first Vice President and one of the founding members of Eco-Tech Empire, a club established in Manipal University Jaipur which incorporates the principles of Green Computing, Cyber Security and Research Work.