Round Robin Scheduling:
History and Development of Round Robin Scheduling

1. Early Concepts of Time-sharing Systems (1950s–1960s)

The **Round Robin (RR) scheduling algorithm** emerged as a natural solution to manage **time-sharing systems**, a concept first explored in the 1950s and 1960s when computers were incredibly expensive and scarce. During this time, multiple users would share a single machine, making it crucial to ensure that each user got a fair share of the machine's computational resources.

- **MIT's Compatible Time-Sharing System (CTSS)**, developed in the early 1960s, was one of the first practical time-sharing systems. The concept of sharing CPU time in slices across multiple tasks can be seen as a precursor to Round Robin.
- In the late 1960s, **John McCarthy**, a computer scientist and pioneer in artificial intelligence, proposed the idea of **time-sharing** for interactive computing. This idea provided the groundwork for RR, which allocates fixed time slices to processes, ensuring responsiveness for interactive users.

---

2. Development of Multitasking and Preemption (1960s–1970s)

As computing evolved into the 1960s, researchers began developing operating systems capable of running **multiple programs concurrently**. The shift from batch processing to **multitasking** required scheduling algorithms that could share the CPU between multiple processes.

- **Multics (1965–1970)**, a highly influential time-sharing operating system, played a crucial role in advancing the concepts of multitasking. Though not a strict implementation of RR, its emphasis on fairness and sharing of resources influenced the development of preemptive scheduling algorithms, including RR.
- In 1967, **Corbato's Law** proposed that the performance of time-sharing systems, including those based on Round Robin, could be improved by keeping time slices short, which emphasized the importance of fine-tuning the **time quantum**.

---

3. Formalization of Round Robin (1970s)

The RR algorithm became formalized as a distinct **scheduling algorithm** in the 1970s, primarily in **academic discussions on operating systems**. As computers evolved to handle more concurrent users and tasks, RR was identified as an efficient way to manage **CPU time** among multiple processes.

- **Round Robin's core idea**—assigning each process a fixed time quantum before switching to the next—was solidified as a foundational method for ensuring fairness in time-sharing systems.

- The introduction of **preemption** (the ability to interrupt and switch between tasks) became a crucial part of RR. This allowed systems to switch between processes based on time slices, providing greater flexibility and responsiveness compared to earlier non-preemptive scheduling methods.

---

4. Adoption in Modern Operating Systems (1980s–2000s)

By the 1980s and 1990s, the adoption of RR in mainstream operating systems became common, particularly in systems designed for interactive users. As **graphical user interfaces (GUIs)** became popular, the need for responsive multitasking systems solidified RR as a vital scheduling algorithm.

- **Unix** and its derivatives (such as **Linux**) adopted **time-sharing** and **preemptive multitasking**, where variations of RR scheduling were often employed. Unix-like systems use RR for interactive processes, ensuring responsiveness.
- **Windows NT**, introduced in the early 1990s, also implemented variations of RR for certain types of processes in its multitasking environment, ensuring responsiveness in desktop applications.

---

5. Advances in Hybrid Scheduling (2000s–Present)

While Round Robin remains a simple and effective algorithm, modern systems have developed more complex and adaptive scheduling methods. However, RR still plays an essential role, especially for **interactive systems** and **real-time applications**.

- **Modern Linux kernels**, for instance, employ a more sophisticated hybrid scheduling algorithm (Completely Fair Scheduler or CFS), but RR is still used in specific cases, particularly in **real-time tasks**.
- In **cloud computing** and **virtualization**, RR remains crucial for **load balancing** and **resource allocation**. Techniques such as **weighted Round Robin** are used to ensure that tasks are scheduled according to their priority or importance while maintaining fairness.
- In **networking**, Round Robin algorithms have evolved into advanced forms, such as **Deficit Round Robin (DRR)** and **Weighted Fair Queuing (WFQ)**, to handle varying data packet sizes more efficiently while maintaining fairness across data streams.