CPU-Scheduling-Algorithms

An implementation of various CPU scheduling algorithms in C++. The algorithms included are First Come First Serve (FCFS), Round Robin (RR), Shortest Process Next (SPN), Shortest Remaining Time (SRT), Highest Response Ratio Next (HRRN), Feedback (FB) and Aging.

## Table of Contents

## Algorithms

### **First Come First Serve (FCFS)**
- First Come First Served (FCFS) is a scheduling algorithm in which the process that arrives first is executed first. It is a simple and easy-to-understand algorithm, but it can lead to poor performance if there are processes with long burst times. This algorithm does not have any mechanism for prioritizing processes, so it is considered a non-preemptive algorithm. In FCFS scheduling, the process that arrives first is executed first, regardless of its burst time or priority. This can lead to poor performance, as longer running processes will block shorter ones from being executed. It is commonly used in batch systems where the order of the processes is important.

### **Round Robin with varying time quantum (RR)**
- Round Robin (RR) with variable quantum is a scheduling algorithm that uses a time-sharing approach to divide CPU time among processes. In this version of RR, the quantum (time slice) is not fixed and can be adjusted depending on the requirements of the processes. This allows processes with shorter burst times to be given smaller quanta and vice versa.

- The algorithm works by maintaining a queue of processes, where each process is given a quantum of time to execute on the CPU. When a process's quantum expires, it is moved to the back of the queue, and the next process in the queue is given a quantum of time to execute.

- The variable quantum allows the algorithm to be more efficient as it allows the CPU to spend more time on shorter processes and less time on longer ones. This can help to minimize the average waiting time for processes. Additionally, it also helps to avoid the issue of starvation, which occurs when a process with a long burst time prevents other processes from executing.

### **Shortest Process Next (SPN)**
- Shortest Process Next (SPN) is a scheduling algorithm that prioritizes the execution of processes based on their burst time, or the amount of time they need to complete their task. It is a non-preemptive algorithm which means that once a process starts executing, it runs until completion or until it enters a waiting state.

- The algorithm maintains a queue of processes, where each process is given a burst time when it arrives. The process with the shortest burst time is executed first, and as new processes arrive, they are added to the queue and sorted based on their burst time. The process with the shortest burst time will always be at the front of the queue, and thus will always be executed next.

- This algorithm can be beneficial in situations where the objective is to minimize the average waiting time for processes, since shorter processes will be executed first, and thus will spend less time waiting in the queue. However, it can lead to longer running processes being blocked by shorter ones, which can negatively impact the overall performance of the system.

- In summary, SPN is a scheduling algorithm that prioritizes the execution of processes based on their burst time, it's non-preemptive and it's commonly used in situations where the objective is to minimize the average waiting time for processes.

### Shortest Remaining Time (SRT)
- Shortest Remaining Time Next (SRT) is a scheduling algorithm that is similar to the Shortest Process Next (SPN) algorithm, but it is a preemptive algorithm. This means that once a process starts executing, it can be interrupted by a new process with a shorter remaining time.

- The algorithm maintains a queue of processes, where each process is given a burst time when it arrives. The process with the shortest remaining time is executed first, and as new processes arrive, they are added to the queue and sorted based on their remaining time. The process with the shortest remaining time will always be at the front of the queue, and thus will always be executed next.

- This algorithm can be beneficial in situations where the objective is to minimize the average waiting time for processes, since shorter processes will be executed first, and thus will spend less time waiting in the queue. Additionally, it can be useful in situations where the burst time of processes is not known in advance, since the algorithm can adapt to changes in the remaining time as the process is executing.

- In summary, SRT is a scheduling algorithm that prioritizes the execution of processes based on their remaining time, it's a preemptive algorithm, which means that it can interrupt a process that's already executing if a new process with a shorter remaining time arrives and it's commonly used in situations where the objective is to minimize the average waiting time for processes and burst time is not known in advance.

### Highest Response Ratio Next (HRRN)

- Highest Response Ratio Next (HRRN) is a scheduling algorithm that prioritizes the execution of processes based on their response ratio. It is a non-preemptive algorithm which means that once a process starts executing, it runs until completion or until it enters a waiting state.

- The response ratio is calculated by taking the ratio of the waiting time of a process and its burst time. The process with the highest response ratio is executed first, and as new processes arrive, they are added to the queue and sorted based on their response ratio. The process with the highest response ratio will always be at the front of the queue, and thus will always be executed next.

- This algorithm can be beneficial in situations where the objective is to minimize the average waiting time for processes, since processes with longer waiting times will be executed first, and thus will spend less time waiting in the queue. Additionally, it can be useful in situations where the burst time of processes is not known in advance, since the algorithm can adapt to changes in the waiting time as the process is executing.

- In summary, HRRN is a scheduling algorithm that prioritizes the execution of processes based on their response ratio, it's non-preemptive and it's commonly used in situations where the objective is to minimize the average waiting time for processes and burst time is not known in advance.

### Feedback (FB)

- Feedback is a scheduling algorithm that allocates CPU time to different processes based on their priority level. It is a multi-level priority algorithm that uses multiple priority queues, each with a different priority level.

- Processes with higher priority levels are executed first, and as new processes arrive, they are added to the appropriate priority queue based on their priority level. When a process completes execution, it is moved to the next lower priority queue.

- This algorithm can be beneficial in situations where the system needs to handle a mix of short and long-running processes, as well as processes with varying priority levels. By having multiple priority queues, it ensures that processes with higher priority levels are executed first, while also allowing lower-priority processes to eventually be executed.

- In summary, Feedback is a scheduling algorithm that allocates CPU time based on priority levels, it uses multiple priority queues with different levels of priority, processes with higher priority levels are executed first and when process completes execution, it is moved to the next lower priority queue, it's commonly used in situations where system needs to handle a mix of short and long-running processes, as well as processes with varying priority levels.

### Feedback with varying time quantum (FBV)

- Same as [Feedback](#feedback-fb) but with varying time quantum.
- Feedback with varying time quantum also uses multiple priority queues and assigns a different time quantum for each priority level, it allows the algorithm to be more efficient by spending more time on higher-priority processes and less time on lower-priority processes.

### Aging

- Xinu is an operating system developed at Purdue University. The scheduling invariant in Xinu assumes that at any
time, the highest priority process eligible for CPU service is executing, with round-robin scheduling for processes of
equal priority. Under this scheduling policy, the processes with the highest priority will always be executing. As a
result, all the processes with lower priority will never get CPU time. As a result, starvation is produced in Xinu when
we have two or more processes eligible for execution that have different priorities. For ease of discussion, we call the
set of processes in the ready list and the current process as the eligible processes.

- To overcome starvation, an aging scheduler may be used. On each rescheduling operation, a timeout for instance, the
scheduler increases the priority of all the ready processes by a constant number. This avoids starvation as each ready
process can be passed over by the scheduler only a finite number of times before it has the highest priority.

- Each process has an initial priority that is assigned to it at process creation. Every time the scheduler is called it takes
the following steps.
    - The priority of the current process is set to the initial priority assigned to it.
    - The priorities of all the ready processes (not the current process) are incremented by 1.
    - The scheduler choses the highest priority process from among all the eligible processes.

- Note that during each call to the scheduler, the complete ready list has to be traversed.

## Input Format
- Line 1: "trace" or "stats"
- Line 2: a comma-separated list telling which CPU scheduling policies to be analyzed/visualized along with
their parameters, if applicable. Each algorithm is represented by a number as listed in the introduction section and as shown in the attached testcases.
Round Robin and Aging have a parameter specifying the quantum q to be used. Therefore, a policy
entered as 2-4 means Round Robin with q=4. Also, policy 8-1 means Aging with q=1.
 1. FCFS (First Come First Serve)
 2. RR (Round Robin)
 3. SPN (Shortest Process Next)
 4. SRT (Shortest Remaining Time)

5. HRRN (Highest Response Ratio Next)
6. FB-1, (Feedback where all queues have q=1)
7. FB-2i, (Feedback where q= 2i)
8. Aging

- Line 3: An integer specifying the last instant to be used in your simulation and to be shown on the timeline.
- Line 4: An integer specifying the number of processes to be simulated.
- Line 5: Start of description of processes. Each process is to be described on a separate line. For algorithms 1 through 7, each process is described using a comma-separated list specifying:

  1- String specifying a process name\
  2- Arrival Time\
  3- Service Time

- **Note:** For Aging algorithm (algorithm 8), each process is described using a comma-separated list specifying:

  1- String specifying a process name\
  2- Arrival Time\
  3- Priority
- Processes are assumed to be sorted based on the arrival time. If two processes have the same arrival time, then the one with the lower priority is assumed to arrive first.