Name: Hieu Tran

Course: COMP 354: Operating Systems

Instructor: Shikha Mittal

**Project 2 – Short Term Scheduling**

**Documentation**

**DESCRIPTION**

The program represents the implementation of two CPU scheduling policies, Shortest Process Next (SPN) and Shortest Remaining Time (SRT). SPN is a non-preemptive scheduling policy, in which the process with the shortest service time is selected to run first. On the other hand, SRT is a preemptive scheduling policy, in which the process with the shortest remaining service time is selected, but the running process can be moved back to the Ready state in favor of another process with a shorter remaining service time. The program first reads the list of processes from the input file “input.dat”, then uses either the SPN or SRT algorithm to schedule all processes, and finally write the scheduling results on the output file “output.dat” and print out the global average turnaround time, global average normalized turnaround time and global average waiting time across all processes on the screen.

**ALGORITHMS**

Before going into details about the implementation of the two scheduling algorithms, we should discuss about the structure of a process for this program. In the input file, a process is represented by a process ID (which is unique and indicated by the line number), an arrival time (which indicates the time a process enters the Ready state) and a service time (which is the duration required by a process to complete its execution). However, for the SRT scheduling algorithm, we will also need a remaining time for each process. This remaining time is initially the same as the service time, but it decreases as the process is in the Running state. Finally, when the remaining time of a process becomes 0, it means that process has finished its execution. The input for both scheduling algorithms will be a list of such processes.

Additionally, both scheduling algorithms have to make use of a ready queue. This is the queue used to store the processes that have already arrived and are in the Ready state, waiting to be dispatched. Since both scheduling algorithms assign different priority to different processes, the ready always need to be sorted, by service time for SPN and by remaining time for SRT. However, we can also notice that, since remaining time is not needed for the SPN algorithm, it is always the same as service time. Therefore, for simplicity, we can sort the ready queue by remaining time for both algorithms during implementation.

**a. Shortest Process Next (SPN):**

The implementation of the SPN algorithm is straightforward. We always select the process with the shortest service time to execute next, and since SPN is non-preemptive, the selected process will be running until it completes its execution. The detailed implementation of this algorithm is as followed:

1. Set the current time to the arrival time of the earliest process.
2. Add all processes that have an arrival time earlier than, or the same as, the current time to the ready queue. Sort the queue by service time after each addition.
3. If the ready queue is not empty, dispatch the first process in the ready queue to be the running process. If the ready queue is empty but there is still some process not yet scheduled, go back to step 1. Otherwise, go to step 5.
4. The running process will keep running until it finishes. Change the current time to the time it finishes, write the needed information about the process to the output file, then go back to step 2.
5. Since all processes have been scheduled, compute and print out the global average turnaround time, global average normalized turnaround time and global average waiting time.

**b. Shortest Remaining Time (SRT):**

The implementation of the SRT algorithm is similar to that of the SPN algorithm in some ways. However, since SRT is preemptive, the running process can be moved back to the ready queue if there is another waiting process with a shorter remaining time. Therefore, when a process is running, we have to check if there is another process arriving and if that process has a shorter remaining time than the current running process. The detailed implementation of this algorithm is as followed:

1. Set the current time to the arrival time of the earliest process.
2. Add all processes that have an arrival time the same as the current time to the ready queue. Sort the queue by remaining time after each addition.
3. If there is no running process: If the ready queue is not empty, dispatch the first process in the ready queue to be the running process. If the ready queue is empty but there is still some process not yet scheduled, go back to step 1. Otherwise, go to step 5.
4. If there is a running process:
   1. If the process at the front of the ready queue has a shorter remaining time than the running process, write the needed information about the running process to the output file, then move it back to the ready queue, sort the queue by remaining time and dispatch the first process in the queue.
   2. If there is a process arriving between the current time and the time the running process finishes, set the current time to the arrival time of that process and update the remaining time of the running process. Then, add the arriving process to the ready queue, sort the queue by remaining time and go back to step 4.1.
   3. Else, the running process will keep running until it finishes. Change the current time to the time it finishes, write the needed information about the process to the output file, then go back to step 2.
5. Since all processes have been scheduled, compute and print out the global average turnaround time, global average normalized turnaround time and global average waiting time.

**ANALYSIS**

We can use the following input file below to analyze the two algorithms:



This input file represents a list of 5 processes:

- Process 1 with arrival time of 0 and service time of 3.

- Process 2 with arrival time of 2 and service time of 6.

- Process 3 with arrival time of 4 and service time of 4.

- Process 4 with arrival time of 6 and service time of 5.

- Process 5 with arrival time of 8 and service time of 2.

The two algorithms proceed as followed:

**a. Shortest Process Next (SPN):**

(Note that S refers to service time of a process)

- The current time is set to be 0, the arrival time of P1.

- Add P1 to the ready queue. The ready queue is now:

P1 (S=3)

- Dispatch P1 to be the running process. The ready queue is now empty.

- P1 completes at time 0+3=3, so change the current time to 3. P1 finishes its execution. Write the needed information about P1 to the output file.

- Since P2 arrived at time 2, add P2 to the ready queue. The ready queue is now:

P2 (S=6)

- Dispatch P2 to be the running process. The ready queue is now empty.

- P2 completes at time 3+6=9, so change the current time to 9. P2 finishes its execution. Write the needed information about P2 to the output file.

- Since P3 arrived at time 4, P4 arrived at time 6, and P5 arrived at time 8, add P3, P4 and P5 to the ready queue and sort the queue after each addition. The ready queue is now:

P5 (S=2) 🡪 P3 (S=4) 🡪 P4 (S=5)

- Dispatch P5 to be the running process. The ready queue is now:

P3 (S=4) 🡪 P4 (S=5)

- P5 completes at time 9+2=11, so change the current time to 11. P5 finishes its execution. Write the needed information about P5 to the output file.

- There is no arriving process, so dispatch P3 to be the running process. The ready queue is now:

P4 (S=5)

- P3 completes at time 11+4=15, so change the current time to 15. P3 finishes its execution. Write the needed information about P3 to the output file.

- There is no arriving process, so dispatch P4 to be the running process. The ready queue is now empty.

- P4 completes at time 15+5=20, so change the current time to 20. P4 finishes its execution. Write the needed information about P4 to the output file.

- There is no arriving process and the ready queue is empty, so all processes have been scheduled. Calculate and print out the global averages.

A screenshot of a computer

Description automatically generatedOutput.dat:

A close up of a sign

Description automatically generatedScreen output:

**b. Shortest Remaining Time (SRT):**

(Note that R refers to remaining time of a process)

- The current time is set to be 0, the arrival time of P1.

- Add P1 to the ready queue. The ready queue is now:

P1 (R=3)

- Dispatch P1 to be the running process. The ready queue is now empty.

- At time 2, P2 arrives at the ready queue. The running process is now P1 (R=1). The ready queue is now:

P2 (R=6)

- Since the remaining time of P1 is shorter than that of P2, P1 keeps running and completes at time 2+1=3. Change the current time to 3 and write the needed information about P1 to the output file.

- Dispatch P2 to be the running process. The ready queue is now empty.

- At time 4, P3 arrives at the ready queue. The running process is now P2 (R=5). The ready queue is now:

P3 (R=4)

- Since the remaining time of P3 is shorter than that of P2, write the needed information about P2 to the output file and move it back to the queue. Dispatch P3 to be the running process. The ready queue is now:

P2 (R=5)

- At time 6, P4 arrives at the ready queue. The running process is now P3 (R=2). The ready queue is now:

P2 (R=5) 🡪 P4 (R=5)

- Since the remaining time of P3 is shorter than that of P2, P3 keeps running.

- At time 8, P5 arrives at the ready queue. The running process is now P3 (R=0). The ready queue is now:

P5 (R=2) 🡪 P2 (R=5) 🡪 P4 (R=5)

- P3 completes its execution. Write the needed information about P3 to the output file.

- There is no arriving process, so dispatch P5 to be the running process. The ready queue is now:

P2 (R=5) 🡪 P4 (R=5)

- P5 keeps running and completes at time 8+2=10. Change the current time to 10 and write the needed information about P5 to the output file.

- There is no arriving process, so dispatch P2 to be the running process. The ready queue is now:

P4 (R=5)

- P2 keeps running and completes at time 10+5=15. Change the current time to 15 and write the needed information about P2 to the output file.

- There is no arriving process, so dispatch P4 to be the running process. The ready queue is now empty.

- P4 keeps running and completes at time 15+5=20. Change the current time to 20 and write the needed information about P4 to the output file.

- There is no arriving process and the ready queue is empty, so all processes have been scheduled. Calculate and print out the global averages.

A close up of a sign

Description automatically generatedOutput.dat:

A close up of a sign

Description automatically generatedScreen output: