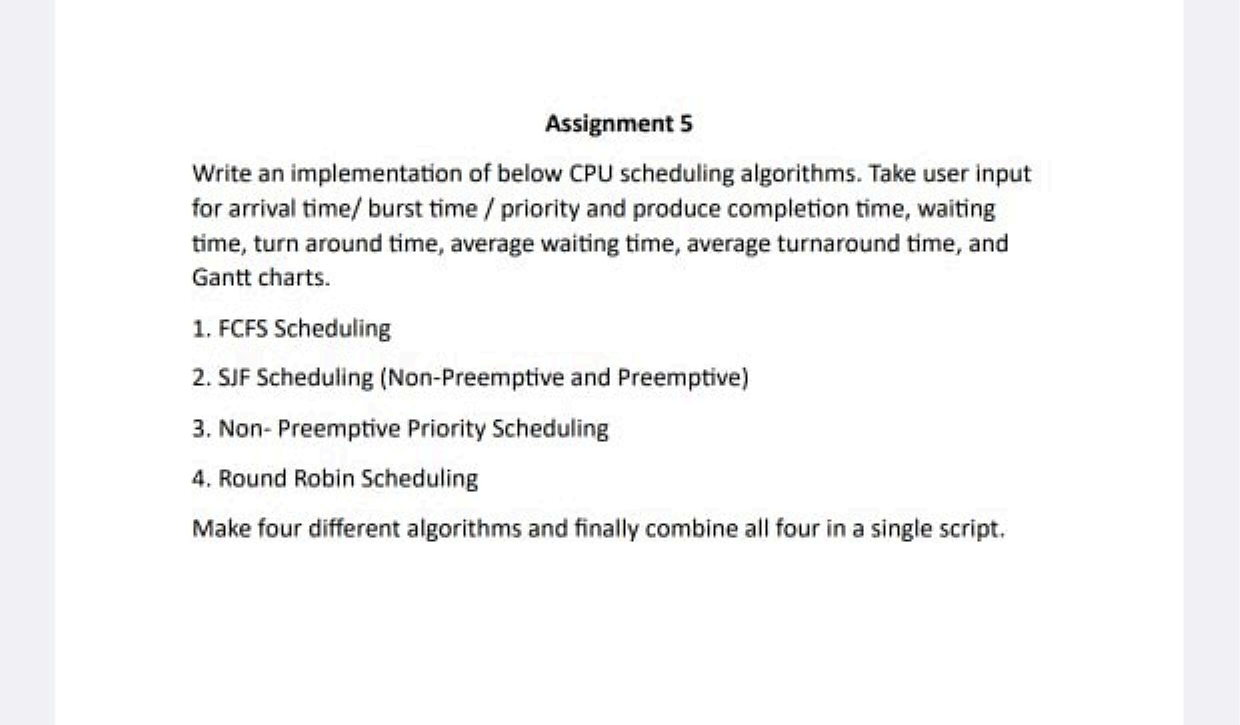
PUNJAB ENGINEERING COLLEGE (DEEMED TO BE UNIVERSITY) CHANDIGARH

Assignment 5

Submitted By :

Sugam Arora
SID : 21105021
Branch : ECE
Date : 29th March, 2025



Assignment 5

Write an implementation of below CPU scheduling algorithms. Take user input for arrival time/ burst time / priority and produce completion time, waiting time, turn around time, average waiting time, average turnaround time, and Gantt charts.

1. FCFS Scheduling

2. SJF Scheduling (Non-Preemptive and Preemptive)

3. Non- Preemptive Priority Scheduling

4. Round Robin Scheduling

Make four different algorithms and finally combine all four in a single script.

Bash Script

```
#!/bin/bash

# Function to perform FCFS Scheduling
fcfs_scheduling() {
    echo -e "\nRunning FCFS Scheduling..."
    # Arrays to store process details
    declare -a at bt ct wt tat

    for ((i=0; i<n; i++)); do
        echo "Process $((i+1)):"
        echo -n "Arrival Time: "
        read at[$i]
        echo -n "Burst Time: "
        read bt[$i]
    done

    # Sort processes by arrival time
    for ((i=0; i<n-1; i++)); do
        for ((j=0; j<n-i-1; j++)); do
            if [ ${at[$j]} -gt ${at[$((j+1))]} ]; then
                # Swap arrival time
                temp=${at[$j]}
                at[$j]=${at[$((j+1))]}
                at[$((j+1))]=$temp

                # Swap burst time
                temp=${bt[$j]}
                bt[$j]=${bt[$((j+1))]}
                bt[$((j+1))]=$temp
            fi
        done
    done

    # Calculate Completion Time, Turnaround Time, and Waiting Time
    ct[0]=$((at[0] + bt[0]))
    tat[0]=$((ct[0] - at[0]))
    wt[0]=$((tat[0] - bt[0]))

    total_wt=${wt[0]}
    total_tat=${tat[0]}

    for ((i=1; i<n; i++)); do
        if [ ${ct[$((i-1))]} -lt ${at[$i]} ]; then
            ct[$i]=$((at[$i] + bt[$i]))
        else
            ct[$i]=$((ct[$((i-1))] + bt[$i]))
        fi
        tat[$i]=$((ct[$i] - at[$i]))
        wt[$i]=$((tat[$i] - bt[$i]))

        total_wt=$((total_wt + wt[$i]))
        total_tat=$((total_tat + tat[$i]))
    done

    avg_wt=$(echo "scale=2; $total_wt / $n" | bc)
    avg_tat=$(echo "scale=2; $total_tat / $n" | bc)

    # Print the results
    echo -e "\nProcess\tAT\tBT\tCT\tWT\tTAT"
    for ((i=0; i<n; i++)); do
        echo -e "P$((i+1))\t${at[$i]}\t${bt[$i]}\t${ct[$i]}\t${wt[$i]}\t${tat[$i]}"
    done
    echo -e "\nAverage Waiting Time: $avg_wt"
    echo -e "Average Turnaround Time: $avg_tat"

    # Gantt Chart
```

```
    echo -e "\nGantt Chart:"
    for ((i=0; i<n; i++)); do
        echo -n "| P$((i+1)) "
    done
    echo "|"
    echo -n "0"
    for ((i=0; i<n; i++)); do
        echo -n "    ${ct[$i]}"
    done
    echo -e "\n"
}

# Function to perform SJF Non-Preemptive Scheduling
sjf_non_preemptive() {
    echo -e "\nRunning SJF Non-Preemptive Scheduling..."
    # Arrays to store process details
    declare -a at bt ct wt tat completed pid gantt gantt_ct

    for ((i=0; i<n; i++)); do
        echo "Process $((i+1)):"
        echo -n "Arrival Time: "
        read at[$i]
        echo -n "Burst Time: "
        read bt[$i]
        pid[$i]=$((i+1))  # Assign process ID
        completed[$i]=0    # Mark as not completed
    done

    time=0  # Current time
    completed_count=0
    total_wt=0
    total_tat=0

    # SJF Non-Preemptive Scheduling
    while [ $completed_count -lt $n ]; do
        # Find process with shortest burst time that has arrived
        min_bt=9999
        min_index=-1

        for ((i=0; i<n; i++)); do
            if [ ${completed[$i]} -eq 0 ] && [ ${at[$i]} -le $time ] && [ ${bt[$i]} -lt $min_bt ]; then
                min_bt=${bt[$i]}
                min_index=$i
            fi
    done

    if [ $min_index -ne -1 ]; then
        # Calculate completion, turnaround, and waiting times
        time=$((time + bt[$min_index]))
        ct[$min_index]=$time
        tat[$min_index]=$((ct[$min_index] - at[$min_index]))
        wt[$min_index]=$((tat[$min_index] - bt[$min_index]))
        total_wt=$((total_wt + wt[$min_index]))
        total_tat=$((total_tat + tat[$min_index]))

        # Mark process as completed
        completed[$min_index]=1
        gantt[$completed_count]=${pid[$min_index]}  # Store process ID for Gantt chart
        gantt_ct[$completed_count]=${ct[$min_index]}  # Store completion time for Gantt chart
        completed_count=$((completed_count + 1))
    else
        time=$((time + 1))  # Increment time if no process is ready
    fi
done

# Calculate averages
avg_wt=$(echo "scale=2; $total_wt / $n" | bc)
avg_tat=$(echo "scale=2; $total_tat / $n" | bc)

# Print the results
echo -e "\nProcess\tAT\tBT\tCT\tWT\tTAT"
for ((i=0; i<n; i++)); do
    echo -e "P${pid[$i]}\t${at[$i]}\t${bt[$i]}\t${ct[$i]}\t${wt[$i]}\t${tat[$i]}"
done
echo -e "\nAverage Waiting Time: $avg_wt"
echo -e "Average Turnaround Time: $avg_tat"

# Gantt Chart
echo -e "\nGantt Chart:"

# Print the Gantt Chart process bar
echo -n "|"
for ((i=0; i<completed_count; i++)); do
    printf " P%-2s |" "${gantt[$i]}"
done
echo

# Print the Gantt Chart time axis (aligned properly)
```

```
    printf "%-4s" "0"  # Start from time 0
    for ((i=0; i<completed_count; i++)); do
        printf "%-6s" "${gantt_ct[$i]}"
    done
    echo -e "\n"
}

# Function to perform Round Robin Scheduling
round_robin() {
    echo -e "\nRunning Round Robin Scheduling..."
    # Arrays to store process details
    declare -a at bt ct wt tat pid remaining_bt

    for ((i=0; i<n; i++)); do
        echo "Process $((i+1)):"
        echo -n "Arrival Time: "
        read at[$i]
        echo -n "Burst Time: "
        read bt[$i]
        pid[$i]=$((i+1))  # Assign process ID
        remaining_bt[$i]=${bt[$i]}  # Initially, remaining burst time = burst time
    done

    echo -n "Enter time quantum: "
    read quantum

    time=0  # Current time
    completed_count=0
    total_wt=0
    total_tat=0
    gantt=()

    # Round Robin Scheduling
    while [ $completed_count -lt $n ]; do
        any_process_left=false  # Flag to check if any process is left to run

        for ((i=0; i<n; i++)); do
            if [ ${remaining_bt[$i]} -gt 0 ] && [ ${at[$i]} -le $time ]; then
                any_process_left=true
                if [ ${remaining_bt[$i]} -le $quantum ]; then
                    time=$((time + remaining_bt[$i]))
                    remaining_bt[$i]=0
                    ct[$i]=$time
                    tat[$i]=$((ct[$i] - at[$i]))
                    wt[$i]=$((tat[$i] - bt[$i]))
                    total_wt=$((total_wt + wt[$i]))
                    total_tat=$((total_tat + tat[$i]))
                    completed_count=$((completed_count + 1))
                else
                    time=$((time + quantum))
                    remaining_bt[$i]=$((remaining_bt[$i] - quantum))
                fi
                gantt+=(${pid[$i]})
            fi
        done

        if [ "$any_process_left" = false ]; then
            break
        fi
    done

    avg_wt=$(echo "scale=2; $total_wt / $n" | bc)
    avg_tat=$(echo "scale=2; $total_tat / $n" | bc)

    # Print the results
    echo -e "\nProcess\tAT\tBT\tCT\tWT\tTAT"
    for ((i=0; i<n; i++)); do
        echo -e "P${pid[$i]}\t${at[$i]}\t${bt[$i]}\t${ct[$i]}\t${wt[$i]}\t${tat[$i]}"
    done
    echo -e "\nAverage Waiting Time: $avg_wt"
    echo -e "Average Turnaround Time: $avg_tat"

    # Gantt Chart
    echo -e "\nGantt Chart:"
    echo -n "|"
    for pid in "${gantt[@]}"; do
        printf " P%-2s |" "$pid"
    done
    echo
    echo -e "\n"
}

# Main Menu to call all functions
echo "Choose Scheduling Algorithm:"
echo "1. FCFS Scheduling"
echo "2. SJF Non-Preemptive Scheduling"
echo "3. SJF Preemptive Scheduling (SRTF)"
echo "4. Non-Preemptive Priority Scheduling"
echo "5. Round Robin Scheduling"
echo "6. Run All Scheduling Algorithms"
read choice

case $choice in
    1) fcfs_scheduling ;;
    2) sjf_non_preemptive ;;
    3) sjf_preemptive ;;
    4) priority_scheduling ;;
    5) round_robin ;;
    6)
        # Running all algorithms for the same input
        echo -n "Enter the number of processes: "
        read n
        echo -e "\nRunning FCFS Scheduling..."
        fcfs_scheduling
        echo -e "\nRunning SJF Non-Preemptive Scheduling..."
        sjf_non_preemptive
        echo -e "\nRunning SJF Preemptive Scheduling..."
        sjf_preemptive
        echo -e "\nRunning Non-Preemptive Priority Scheduling..."
        priority_scheduling
        echo -e "\nRunning Round Robin Scheduling..."
        round_robin
        ;;
    *) echo "Invalid choice" ;;
esac
```