**Lab Exercise – 6**

* AIM ::

WAP in shell script to implement CPU scheduling for `shortest job first` (sjf).

Source\_Code ::

### echo $'\n' "5C6 - Amit Singhal (11614802722)" $'\n'

### read -p "Enter the number of processes: " num\_processes

### echo $'\n' "Enter Arrival Time & Burst Time for $num\_processes processes"

### # Collect process details

### for ((i=0;i<num\_processes;i++)); do

### echo -n "P$((i+1)): "

### read arrival\_time burst\_time

### processes[$i]="$arrival\_time $burst\_time"

### done

### # Initialize variables

### total\_completion\_time=0

### total\_waiting\_time=0

### total\_turnaround\_time=0

### completed\_processes=0

### gantt\_chart="0" # Start Gantt chart at time 0

### time=0

### # Create an array to store completion status of each process (0 = incomplete, 1 = complete)

### for ((i=0;i<num\_processes;i++)); do

### process\_completed[$i]=0

### done

### # Function to find the process with the shortest burst time among those that have arrived

### find\_shortest\_job() {

### local min\_burst=-1

### local min\_index=-1

### for ((i=0;i<num\_processes;i++)); do

### current\_process=(${processes[$i]})

### current\_arrival\_time=${current\_process[0]}

### current\_burst\_time=${current\_process[1]}

### if (( process\_completed[$i] == 0 && current\_arrival\_time <= time )); then

### if (( min\_burst == -1 || current\_burst\_time < min\_burst )); then

### min\_burst=$current\_burst\_time

### min\_index=$i

### fi

### fi

### done

### echo $min\_index

### }

### # Display table header

### echo -e "\nProcess Arrival Time Burst Time Completion Time Turnaround Time Waiting Time"

### # Process all processes using SJF

### while (( completed\_processes < num\_processes )); do

### shortest\_job=$(find\_shortest\_job)

### if (( shortest\_job == -1 )); then

### # No process available, increase time (idle)

### gantt\_chart+=" -- XX -- $((++time))"

### else

### current\_process=(${processes[$shortest\_job]})

### current\_arrival\_time=${current\_process[0]}

### current\_burst\_time=${current\_process[1]}

### if (( time < current\_arrival\_time )); then

### time=$current\_arrival\_time

### gantt\_chart+=" -- XX -- $time"

### fi

### completion\_time=$((time + current\_burst\_time))

### turnaround\_time=$((completion\_time - current\_arrival\_time))

### waiting\_time=$((turnaround\_time - current\_burst\_time))

### # Update total values

### total\_completion\_time=$completion\_time

### total\_waiting\_time=$((total\_waiting\_time + waiting\_time))

### total\_turnaround\_time=$((total\_turnaround\_time + turnaround\_time))

### # Mark the process as completed

### process\_completed[$shortest\_job]=1

### completed\_processes=$((completed\_processes + 1))

### # Display process details

### echo -e "P$((shortest\_job+1))\t\t$current\_arrival\_time\t\t$current\_burst\_time\t\t$completion\_time\t\t $turnaround\_time\t\t $waiting\_time"

### # Update Gantt chart

### gantt\_chart+=" -- P$((shortest\_job+1)) -- $completion\_time"

### # Update current time

### time=$completion\_time

### fi

### done

### # Calculate averages

### avg\_waiting\_time=$(awk "BEGIN {printf \"%.2f\", $total\_waiting\_time/$num\_processes}")

### avg\_turnaround\_time=$(awk "BEGIN {printf \"%.2f\", $total\_turnaround\_time/$num\_processes}")

### # Display Gantt chart

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

### echo -e "$gantt\_chart"

### # Display averages

### echo ""

### echo "Avg waiting time: $avg\_waiting\_time"

### echo "Avg turnaround time: $avg\_turnaround\_time"

### 

Output ::

Output ::