Question – 1

This is a C program that implements the First-Come-First-Serve (FCFS) CPU scheduling algorithm. The program calculates the waiting time, turnaround time and average waiting time and average turnaround time for a given set of processes. The user is prompted to enter the number of processes to be executed, and the burst time and arrival time for each process.

How to use

1. Open a text editor or an integrated development environment (IDE) that supports C programming language such as Visual Studio Code, Dev-C++, or CodeBlocks.

2. Create a new file and copy-paste the code into the file.

3. Save the file with a .c extension (e.g. "cpu-scheduling.c").

4. Compile and run the program using a C compiler such as GCC or Clang.

Program Flow

1. The user is prompted to enter the total number of processes to be executed.

2. The program asks for the burst time and arrival time for each process.

3. The waiting time for each process is calculated.

4. The Gantt chart is displayed.

5. The turnaround time for each process is calculated.

6. The average waiting time and average turnaround time are calculated.

7. The program prints the results.

Variables used

• n: The total number of processes to be executed.

• i: A loop variable used to iterate through the processes.

• awt: The average waiting time.

• att: The average turnaround time.

• btime[]: An array that stores the burst time for each process.

• wtime[]: An array that stores the waiting time for each process.

• ttime[]: An array that stores the turnaround time for each process.

• atime[]: An array that stores the arrival time for each process.

Question – 2

This program implements a basic scheduling algorithm called Shortest Job First (SJF) that schedules processes based on their burst time, where the process with the shortest burst time is scheduled first. The program takes in user input for the number of processes, their respective burst times and arrival times. It then sorts the processes by their burst times and calculates the waiting time, turnaround time and generates a Gantt chart to visualize the execution order.

Compilation

To compile the program, run the following command in the terminal:

gcc -o program sjf.c

This will generate an executable file called "program".

Usage

To run the program, execute the following command in the terminal:

./program

The program will then prompt the user to enter the number of processes, followed by their respective burst times and arrival times. After processing the input, the program will output the Gantt chart, waiting time, turnaround time and average waiting and turnaround time.

Inputs

• Number of processes (integer)

• Burst time for each process (array of integers)

• Arrival time for each process (array of integers)

Outputs

• Gantt chart

• Waiting time for each process

• Turnaround time for each process

• Average waiting time

• Average turnaround time

Question – 3

Round Robin Scheduling Algorithm

This program implements the Round Robin scheduling algorithm for a set of processes. Round Robin is a preemptive scheduling algorithm in which each process is executed for a fixed time slice, and then it is preempted to allow other processes to run. The program accepts the number of processes, burst time for each process, and the time quantum from the user. It then calculates the waiting time and turnaround time for each process and displays the results along with the average waiting time and average turnaround time.

Program Description

• The program first accepts the number of processes and time quantum from the user.

• It then accepts the burst time for each process and initializes the remaining time for each process to its burst time.

• The program then runs a loop until all processes are completed. Inside the loop, it runs another loop for each process and checks if the remaining time for the process is greater than 0.

• If the remaining time for the process is less than or equal to the time quantum, the program adds the remaining time to the current time and sets the remaining time to 0 for that process.

• If the remaining time for the process is greater than the time quantum, the program adds the time quantum to the current time and subtracts the time quantum from the remaining time for that process.

• The program then adds the time quantum to the waiting time for all processes except the current process.

• After all processes are completed, the program calculates the waiting time and turnaround time for each process and displays the results along with the average waiting time and average turnaround time.

Program Logic

1. Accept the number of processes and time quantum from the user.

2. Accept the burst time for each process and initialize the remaining time for each process to its burst time.

3. Run a loop until all processes are completed:

1. For each process, check if the remaining time is greater than 0.

2. If the remaining time is less than or equal to the time quantum, add the remaining time to the current time and set the remaining time to 0 for that process.

3. If the remaining time is greater than the time quantum, add the time quantum to the current time and subtract the time quantum from the remaining time for that process.

4. Add the time quantum to the waiting time for all processes except the current process.

4. Calculate the waiting time and turnaround time for each process.

5. Calculate the average waiting time and average turnaround time.

6. Display the results.

How to Use

1. Compile the program using a C compiler.

2. Run the executable file.

3. Enter the number of processes and time quantum when prompted.

4. Enter the burst time for each process when prompted.

5. The program will display the waiting time and turnaround time for each process along with the average waiting time and average turnaround time.

Question – 4

This program implements a Priority Scheduling algorithm to schedule a set of processes. The Priority Scheduling algorithm is a non-preemptive scheduling algorithm, which means that a process will continue to run until it completes its execution or is blocked by an I/O operation.

The program prompts the user to enter the number of processes and the burst time and priority for each process. The program then sorts the processes based on their priorities, and calculates the waiting time and turnaround time for each process. Finally, the program displays the results, including the average waiting time and average turnaround time.

The program uses the following variables:

• num\_process: an integer variable to store the number of processes entered by the user.

• process\_id: an array of integers to store the ID of each process.

• burst\_time: an array of integers to store the burst time of each process.

• priority: an array of integers to store the priority of each process.

• waiting\_time: an array of integers to store the waiting time of each process.

• turnaround\_time: an array of integers to store the turnaround time of each process.

• i, j: integer variables used as counters in the for-loops.

• min\_priority: an integer variable used to store the minimum priority value found during the sorting process.

• min\_priority\_index: an integer variable used to store the index of the process with the minimum priority value found during the sorting process.

• temp\_burst\_time, temp\_priority, temp\_process\_id: integer variables used to temporarily store values during the sorting process.

• avg\_waiting\_time: a float variable used to store the average waiting time of all processes.

• avg\_turnaround\_time: a float variable used to store the average turnaround time of all processes.

The program first prompts the user to enter the number of processes and the burst time and priority for each process. It then uses a selection sort algorithm to sort the processes based on their priorities. The selection sort algorithm works by finding the process with the minimum priority value and swapping it with the process at the current position in the array. The program uses three arrays (burst\_time, priority, and process\_id) to sort the processes simultaneously.

After sorting the processes, the program calculates the waiting time and turnaround time for each process using the formulas:

• Waiting time = (Waiting time of previous process) + (Burst time of previous process)

• Turnaround time = Waiting time + Burst time

The program then calculates the average waiting time and average turnaround time by summing up the waiting time and turnaround time of all processes and dividing by the number of processes.

Finally, the program displays the results, including the process ID, burst time, priority, waiting time, and turnaround time for each process, as well as the average waiting time and average turnaround time.

To run the program, compile the code using a C compiler and execute the resulting executable file. Follow the prompts to enter the number of processes and the burst time and priority for each process. The program will then display the results.