Project Name: Round Robin Scheduler using Circular Doubly Linked List

Author:
My (Jenny) Tran
Date of Completion:
Feb 23, 2024

Code Description

Process Class

Description*:* This class represents a process in the Round Robin Scheduler. Each process has a name (processName) and the total time it takes to run (totalTime)

Public Member Variables

string processName: Represent the name of the process
int totalTime: Represents the total time required for the process to complete

Constructors

Process(string processName, int totalTime): Initializes a process with the given name and total time.

Public Member Functions

void updateRunTime(int qTime)

- Purpose: Updates the remaining time of the process after each quantum cycle
- Precondition: The process isinitialized with valid totalTime
- Postcondition: The totalTime of the process is reduced by the provided qTime

void print()

- Purpose: Print the details of the process
- Precondition: The process has valid processName and totalTime
- Postcondition: Details of the process are printed to the console

int getTime()

- Purpose: Returns the remaining time of the process
- Precondition: The process has been initialized
- Postcondition: The remaining time of the process is returned

Node Class

Description: This class represents a node in the circular doubly linked list. It holds a pointer to the data (Process) and pointers to the next and previous nodes

Public Member Variables

T *data: Pointer to the data (Process object)
Node<T> *next: Pointer to the next node
Node<T> *prev: Pointer to the previous node

Constructors

Node(T *data): Initialize a node with the given data

Public Member Functions

void print()

- Purpose: Print the data of the node
- Precondition: The node has valid data
- Postcondition: Data of the node is printed to the console

CircularDLL Class

Description: This class represents the circular doubly linked list. It is a container for nodes, and it includes operations for inserting and deleting processes

Private Member Variables

Node<T> *head: Pointer to the head of the list
Node<T> *tail: Pointer to the tail of the list
int length: Represents the length of the list

Constructors

CircularDLL(T *data): Initialize the list with a given data

Destructor

~CircularDLL(): Destroy the list and frees memory

Public Member Functions

int getLength()

- Purpose: Return the length of the list
- Precondition: The list has been initialized
- Postcondition: The length of the list is returned

Node<T> *get(int index)

- Purpose: Returns the node at the specified index
- Precondition: The list has been initialized and the index is not out of bound
- Postcondition: The node at the specified index is returned

void printList()

- Purpose: Prints the processes in the list
- Precondition: The list has been initialized
- Postcondition: Details of the processes in the list are printed to the console

void insertProcess(T *data)

- Purpose: Inserts a new process at the end of the list
- Precondition: The list has been initialized, and the new process data is valid
- Postcondition: The new process is inserted at the end of the list

void deleteProcess(int index)

- Purpose: Deletes a process at the specified index
- Precondition: The list has been initialized and the index is valid
- Postcondition: The process at the specified index is deleted from the list

Main Program

***Description**:* The main program initializes a circular doubly linked list with prepopulated processes, runs the Round Robin Scheduler and allow the user to add new processes during the scheduler cycles