Steal job scheduling algorithm

- ## Project specification:

  - ### General description

A steal job scheduling algorithm is an algorithm that is used to do task scheduling for various processors in a device, all processors are stored in a Linked List. Each processor has a list of threads or processes to be executed (deque). The processor gets the first element from the queue to execute the next process. When one process forks its moved back to the front of the next queue and a new thread started execution. If one processor executed all processes it has, it goes to steal a process from another processor and usually it takes the last element of the queue of the other processor to execute it.

Fork happens when one of the processes in a particular queue takes time more than expected so it will be removed from its queue and inserted in the anther queue which has the smallest execution time.

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers.

A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO).

In this project we will use linked list and linked queue with more features added to them which is the ability to insert in the front and the ability to delete from the rear in the linked queue ,its node will be storing information which is the process id and the process execution time, the node of the linked list we be storing pointer of type queue instead of the basic data in the basic linked list, there for we will use nested data structure which is linked list stores queues.

Basic functions in implementation level

- **CreateQueue_q**

  This function is to create a queue and make it in the initialize state , it takes a pointer of a variable of type queue as a parameter and it returns nothing.

- **IsFull_q**

  This function is to check if the queue is full or not, although it will not be full as we are use dynamic allocation, it takes a pointer of a variable of type queue as a parameter.

- **IsEmpty_q**

  This function is to check if the queue is empty or not, it takes a pointer of a variable of type queue as a parameter and it return one if it is empty otherwise it return zero.

- **Append_q**

  This function is to insert in the queue in the rear, it takes process id ,execution time and a pointer of a variable of type queue as a parameter and it returns nothing.

- **Append_f**

  This function is to insert in the queue in the front, it takes process id ,execution time and a pointer of a variable of type queue as a parameter and it returns nothing.

- **Serve_q**

  This function is to get the content of a particular queue if needed, it takes 3 pointers as parameters the first one is to store the process id the second one is to store the execution time and the last one is a pointer of queue that you want to serve from and it returns nothing.

- **QueueSize_q**

  This function is to get the size of a particular queue, it takes a pointer of a variable of type queue as a parameter and it returns the size of the queue.

- **ClearQueue_q**

  This function is to delete all the content of a particular queue from the memory, it takes a pointer of a variable of type queue as a parameter and it returns nothing.

- **LastNode**

  This function is to get the last node of a particular queue, it takes a pointer of a variable of type queue as a parameter and it returns the last node of the queue.

- **CreateList**

  This function is to create a list and make it in the initialize state, it takes a pointer of a variable of type List as a parameter and it returns nothing.

- **ListEmpty**

  This function is to check if the list is empty or not, it takes a pointer of a variable of type list as a parameter and it return one if it is empty otherwise it returns zero.

- **ListFull**

  This function is to check if the list is full or not, although it will not be full as we are use dynamic allocation, it takes a pointer of a variable of type list as a parameter.

- **ListSize**

  This function is to get the size of a particular list, it takes a pointer of a variable of type list as a parameter and it returns the size of the list.

- **InsertList**

  This function is to insert in the list, it takes process id ,execution time and a pointer of a variable of type queue as a parameter and it returns nothing.

- **FindMin_Q**

  This function is to find the minimum of totals execution time of processes in particular queue, it takes a pointer of a variable of type list as a parameter and it returns the address of the queue which has minimum total execution time.

- **FindMax_Q**

  This function is to find the minimum of total execution time of processes in particular queue, it takes a pointer of a variable of type list as a parameter and it returns the address of the queue which has maximum total execution time.

- **Steal**

  This function is to take a process from the queue that has the maximum total execution time and delete it from its queue and insert it in the queue that has the minimum total execution time, it takes a pointer of a variable of type list as a parameter and it returns nothing.

- **Fork**

- **Execute**

  This function is to make our processors work, it takes a pointer of a variable of type list as a parameter and it returns nothing.

- **Display**

  This function is to print the content of a particular list, it takes a pointer of a variable of type list as a parameter and it returns nothing but it will print in the console.

- **DestroyList**

  This function is to delete and deallocate the content of a particular list, it takes a pointer of a variable of type list as a parameter and it returns nothing.

Basic functions in user level

There is no functions in the user level.

- ## Project design and implementation

  - ### Imported libraries and designed algorithms

    - **Libraries**

      - Stdio.h
      - Stdlib.h
      - queue.h
      - List_q.h

    - **Designed algorithms**

      - QueueNode

        This structure consists of three members which are the *process id*, *execution time* and *next* a pointer of type queue node to link the nodes with each other.

      - Queue

        This structure consists of five members which are *front* that is a pointer of type queue node to keep track of the first element inserted to the queue, *rear* that is a pointer of type queue node to keep track of the last element inserted to the queue, *Time_Q*, *totalof_T* which is the total execution time of a queue and *size* which is the number of nodes created inside a queue.

      - ListNode

        This structure consists of two members which are q which is a pointer of type queue as information and *next* a pointer of type List node to link the nodes with each other.

      - List

        This structure consists of two members which are *head* that is a pointer of type list node to keep track of the first element inserted to the queue and *size* which is the number of nodes of type list node created inside a List.