**Assignment\_5(OS)**

Readme

This code provides a simple implementation of parallel loops for 1D and 2D using threads in c++.It contains demonstration functions for passing lambda functions as parameters . This is responsible for performing given parallelising task though multithreading .There are two different function parrallel\_for for 1D and 2D and also contains other necessary function described here :

**Functions:**

1.Parallel\_for 1D: This function takes a low index ,high index,a lambda function and the number of threads as parameters.It Divides the range into chunks and assigns each chunk to a thread for parallel execution .

* Thread Creation and Task Assignment:
  + The function creates a vector thread vector to store thread handles.
  + It iterates over the specified number of threads.
  + For each thread, it calculates a chunk size based on the total range and the number of threads.
  + It determines the start and end indices for the thread's chunk.
  + It creates a tuple containing the start and end indices and the lambda function and passes it as an argument to the thread function.

Thread Execution:

* + It is responsible for executing the lambda function over the 1D range assigned to each thread.
  + The threads run in parallel, working on their assigned chunks of the 1D range.
* Thread Joining:
  + After the threads complete their tasks, the main thread waits for all threads to finish using pthread\_join
* Execution Time Measurement:
  + The function measures and prints the total execution time.

2.Parallel\_for\_2D: This function takes a low index ,high index for both dimension , lambda function and number of threads.

Thread Creation:

* + The function creates a vector thread vector to store thread handles.
  + It iterates over the specified number of threads(numberThreads).
  + For each thread, it calculates a chunk size for the first dimension based on the total range and the number of threads.
  + It determines the start and end indices for the first dimension based on the chunk size.
  + It creates a tuple containing the start and end indices for both dimensions, the lambda function, and passes it as an argument to thread\_function\_2D.

Thread Execution:

* + It is responsible for executing the lambda function over the 2D range assigned to each thread.
  + The threads run in parallel, working on their assigned chunks of the 2D range.
* Thread Joining:
  + After the threads complete their tasks, the main thread waits for all threads to finish using pthread\_join.
* Execution Time Measurement:
  + The function measures and prints the total execution time.

3. Thread\_function\_2D:

It is the thread function that will be executed in parallel. It takes an argument, which is later cast into a tuple containing the parameters needed for the 2D parallel loop and there are two loops which iterates over the first dimension and second dimension of 2D and the call lambda function in 2D range.

4. Thread\_function:

The function first casts the void argument to a tuple containing the start index , end index , and the lambda function .

* Looping Through the 1D Range:
  + The function then iterates through the 1D range.
  + For each index it invokes the provided lambda function.
* Thread Exit:
  + After completing the loop for the assigned 1D range, the thread exits.

**Contributions :**

**Vipul Verma:** thread\_function\_2D , parallel\_for\_2D, Error check and documentation.

**Wasif Ali:** thread\_function ,parallel\_for, Error check

***Git-Hub -*** [*https://github.com/A-WASIF/OS*](https://github.com/A-WASIF/OS)