Project 3

Multithreaded Sorting Application
Fork-Join Sorting Application

Introduction

The goal of this project is to implement multithread algorithms. The three tasks are to check the correctness of a sudoku solution using pthreads, to implement merge sort using pthreads, and to implement merge sort and quick sort using java fork-join framework.

Implementation

Sudoku Solution Checker

The sudoku solution checker is implemented using pthreads. The program reads a sudoku solution from console and checks if the solution is correct. The program creates 11 threads, 1 thread to check rows, 1 thread to check columns, and 9 threads to check each 3x3 subgrid. The program uses a struct to pass the arguments to the threads that check the subgrids.

```
struct CheckSubgridArgs
{
  int (*sudoku)[9];
  int subgrid_row;
  int subgrid_column;
};
```

The threads are created using the following code:

```
  pthread_create(&tid_row, NULL, checkRows, sudoku);
  pthread_create(&tid_column, NULL, checkColumns, sudoku);
  struct CheckSubgridArgs csas[9];
  for (int i = 0; i < 3; ++i)
    for (int j = 0; j < 3; ++j)
    {
      csas[i * 3 + j].sudoku         = sudoku;
      csas[i * 3 + j].subgrid_row    = i;
      csas[i * 3 + j].subgrid_column = j;
      pthread_create(&tid_subgrid[i * 3 + j], NULL,
                     checkSubgrids, &csas[i * 3 + j]);
    }
```

The threads write the result to global variables. The main thread waits for all threads to join and checks the results to determine if the sudoku solution is correct.

Parallel Merge Sort

The parallel merge sort is implemented using pthreads. The program reads a list of integers from console and calls the merge sort function. The merge sort function creates two merge sort threads to sort the left and right halves of the list. The merge sort function waits for the two threads to join and then merges the two sorted halves. The merge sort function is implemented as follows:

```
struct MergeSortArgs
{
  int *array;
  int  begin;
  int  end;
};

void *merge_sort(void *args)
{
  // Parse arguments
  // Create threads for left and right halves
  // Wait for left and right threads to join
  // Create a thread to merge the left and right halves
  pthread_exit(0);
}

struct MergeArgs
{
  int *array;
  int  begin;
  int  middle;
  int  end;
};

void *merge(void *args)
{
  // Parse arguments
  // Create a copy of the left and right halves
  // Merge the left and right halves in the original array
  // Free the copy arrays
  pthread_exit(0);
}
```

Fork-Join Merge Sort and Quick Sort

The fork-join merge sort and quick sort are implemented using the java fork-join framework. The program reads a list of integers from console and calls the merge sort and quick sort functions. Take the quick sort as an example:

The `QuickSort` class create a `ForkJoinPool` and calls the subclass `QuickSortTask` to sort the array. The `QuickSortTask` class extends `RecursiveAction` and overrides the `compute` method to sort the array. The `partition` method is used to partition the array and the `swap` method is used to swap two elements in the array.

In the `compute` method, if the size of the array is less than or equal to the threshold "1", the array with only one element is naturally sorted. Otherwise, the array is partitioned and two `QuickSortTask` instances are created and their `fork` method is called to sort the two partitions. The `join` method is called to wait for the two tasks to complete.

Correctness

Sudoku Solution Checker

The following is the output of the sudoku solution checker program:

```
ceryl@Ceryl:~/os_projects$ ./SudokuCheck
Enter the sudoku:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
The sudoku is valid.
ceryl@Ceryl:~/os_projects$
```

Figure 1: Correct Sudoku Solution

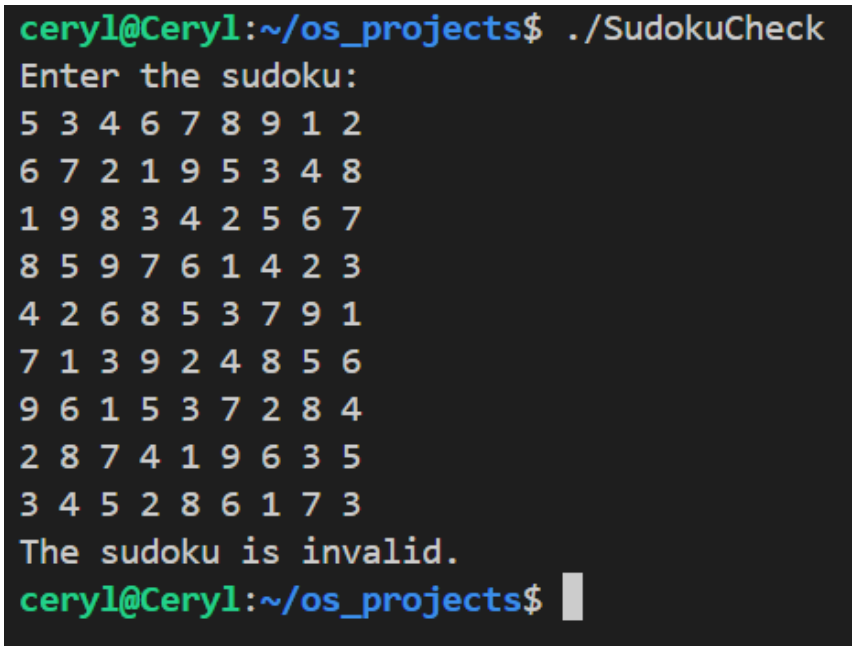

Figure 2: Incorrect Sudoku Solution (at row 9, column 9)

Parallel Merge Sort

The following is the output of the parallel merge sort program:

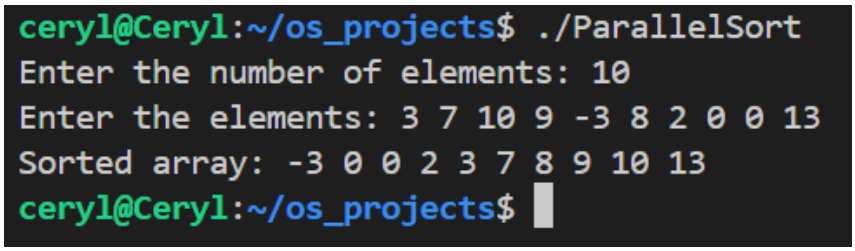

Figure 3: Parallel Merge Sort

Fork-Join Merge Sort and Quick Sort

The two algorithms are tested with a unit test. The following is the output of the unit test:

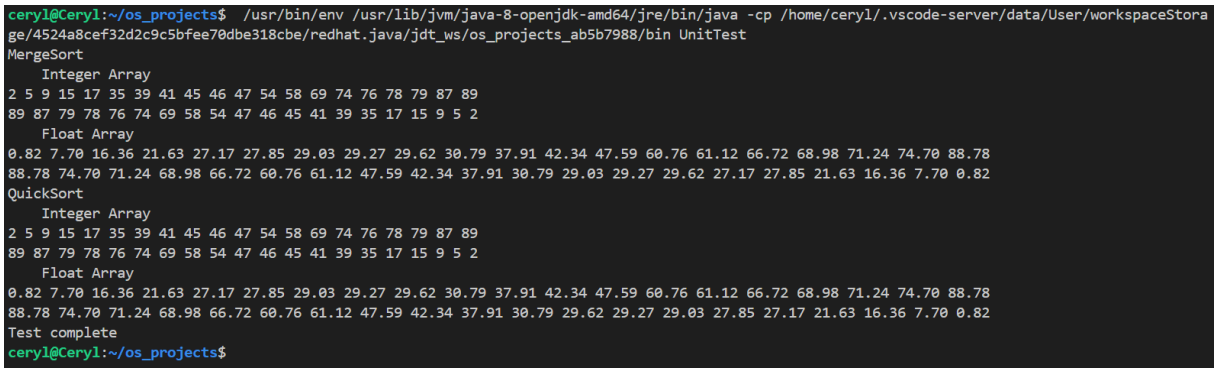

Figure 4: Fork-Join Merge Sort and Quick Sort

Conclusion

The project implements multithread algorithms to do some sorting and checking tasks. The pthread and fork-join frameworks are used to implement the algorithms. The correctness of the algorithms is verified by testing with different inputs.

By implementing the algorithms, I have learned how to use pthreads and fork-join framework to implement multithread algorithms. I have also learned how to use the pthread and fork-join APIs to create and manage threads. The project has helped me to understand the concepts of multithreading and how to implement multithread algorithms in C and Java.