Concurrent Merge Sort

Input: Integer N and N numbers

Functions:

- shareMem: To create a shared memory.
- normal_mergesort: To implement normal merge sort.
- mergesort: To implement merge sort by recursively creating two child processes and merging their output.
- threaded_mergeSort: To implement mergesort by recursively creating two threads and merging their production.
- merge: To combine two sorted arrays.
- Swap: To swap two integers.
- runSorts: To run the three types of mergesorts after creating shared memory.

Explanation:

The program takes input an integer N and N integers. Firstly, a copy of N numbers is sorted by concurrent merge sort. Secondly, another copy of the N numbers is sorted by threaded merge sort, and finally, another copy of numbers is sorted by normal merge sort. During each process for an array of size less than five selection sort is performed. Time taken by each type of merge sort is noted and compared finally.

- In concurrent merge sort, two halves of the array are sorted by two different child process, and the parent process waits for them and merges the two halves.
- In threaded merge sort, two halves of the array are sorted by two different threads, and they are joined, after which parent thread merges the two halves.

Findings:

- Normal merge sort ran around 80-100 times faster compared to concurrent merge sort and about 50 –70 times faster than threaded merge sort.
- Threaded merge sort is faster than Concurrent merge sort as Concurrent merge sort requires duplicating entire data of its parent data on every call of the fork. And it has the overhead of context switch. Threaded merge sort involves overhead of creating threads. So normal merge is fastest of all, threaded merge sort is intermediate, and concurrent merge sort is slowest.