

Department of Computer Science

SPRING 20’

**OPERATING SYSTEM PROJECT**

**SUPERVISED:**

Dr. Hasina Khatoon/Ms. Nausheen Shoaib

Ms. Rahemeen Khan

GROUP MEMBERS

18k-1148 (MIRZA UZAIR ALI)

18K-1268 (HASAN SHAFIQ)

Introduction:

Main purpose of our project is to compare the threads with processes in terms of performance and time taken to complete a task. We have used different sorting algorithms to conclude our results.

Brief Overview:

The time taken at each step is recorded and displayed. At the end, the 4 sub arrays combine into one array and printed into the log file. Before exiting, The program also shows the result that which method (processes or threads) was faster and how much less time it takes than the another.

Modules:

Bubble Sort

Insertion Sort

Selection Sort

Radix Sort

Count Sort

Heap Sort

Implementation:

POSIX Threads:

4 P-threads were created and then joined

Processes:

It was implemented using 2 fork calls and then the if else-if conditions were used to separate different processes and make them work accordingly.

Shared Memory:

We have to use shared memory for implementing Processes so that the sorting done on different sub arrays in different processes should reflect in parent process also.

Header Files:

#include <iostream> : common library for basic stuff

#include <unistd.h>: common library for basic stuff

#include <stdlib.h>: common library for basic stuff

#include <time.h>: we need to seed the random function with time(0)

#include <fstream>: for filing (the log)

#include <chrono>: provides high precision clock so that we can record the time taken by a single or multiple instructions

#include <sys/wait.h>: wait(NULL) support from this library

#include <sys/shm.h>: shared memory support from this library

#include <sys/stat.h>: common library for basic stuff

#include <sys/types.h>: common library for basic stuff

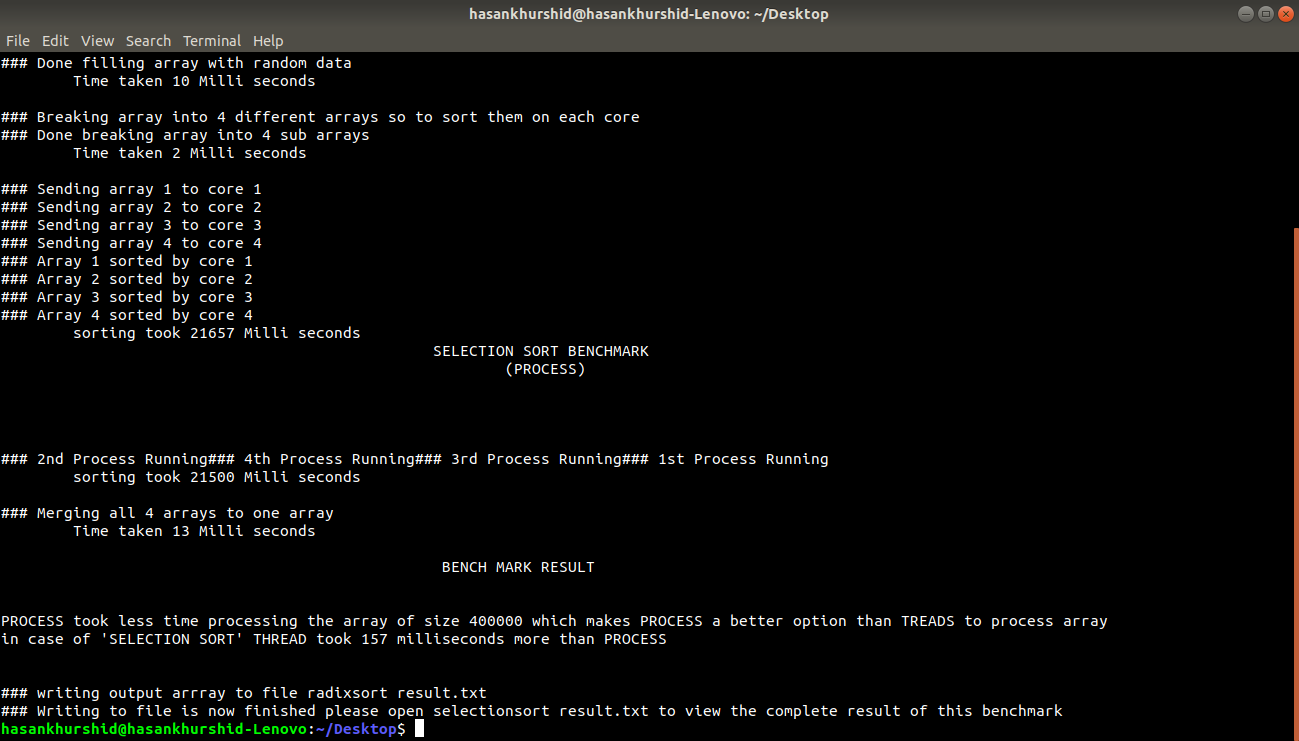
#include <sys/ipc.h>: INTER PROCESS COMUNICATION ipc used because we needed to make a channel through which parent and child could communicate

Conclusion:

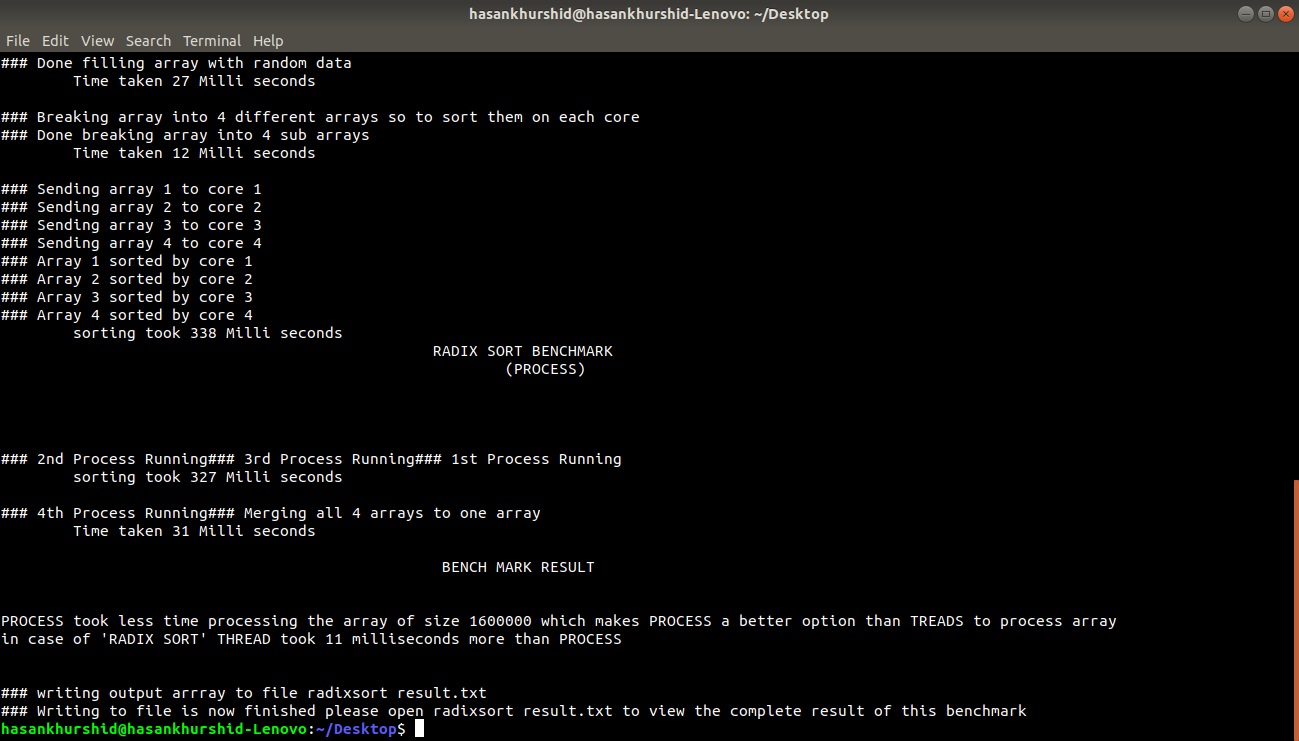
After running all 6 benchmarks on the same machine we found out that the HEAP and COUNT sort sorted the array of size 16,000,000 within 300 milliseconds while other took around 50k milliseconds to sort array of size just 4,000,000 which proves HEAP and COUNT sort better that other 4 sorting algorithms.

Screenshots:

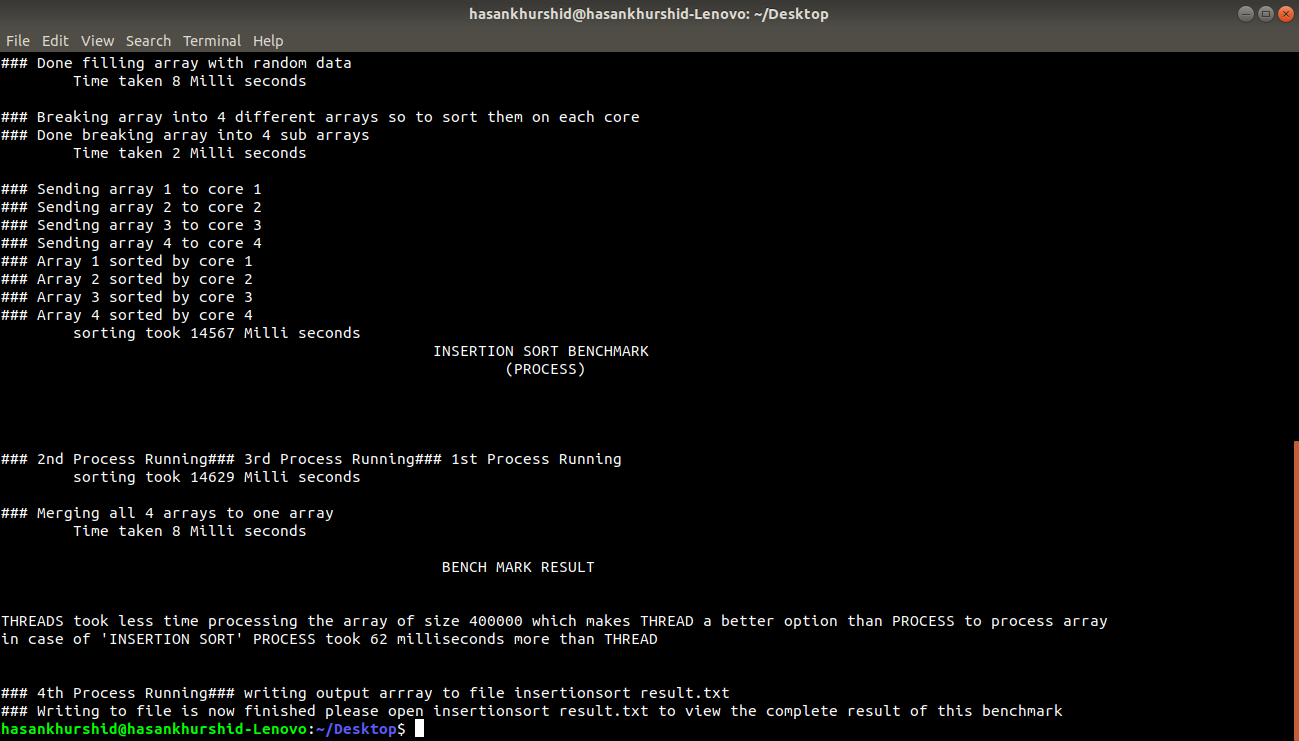
Selection Sort



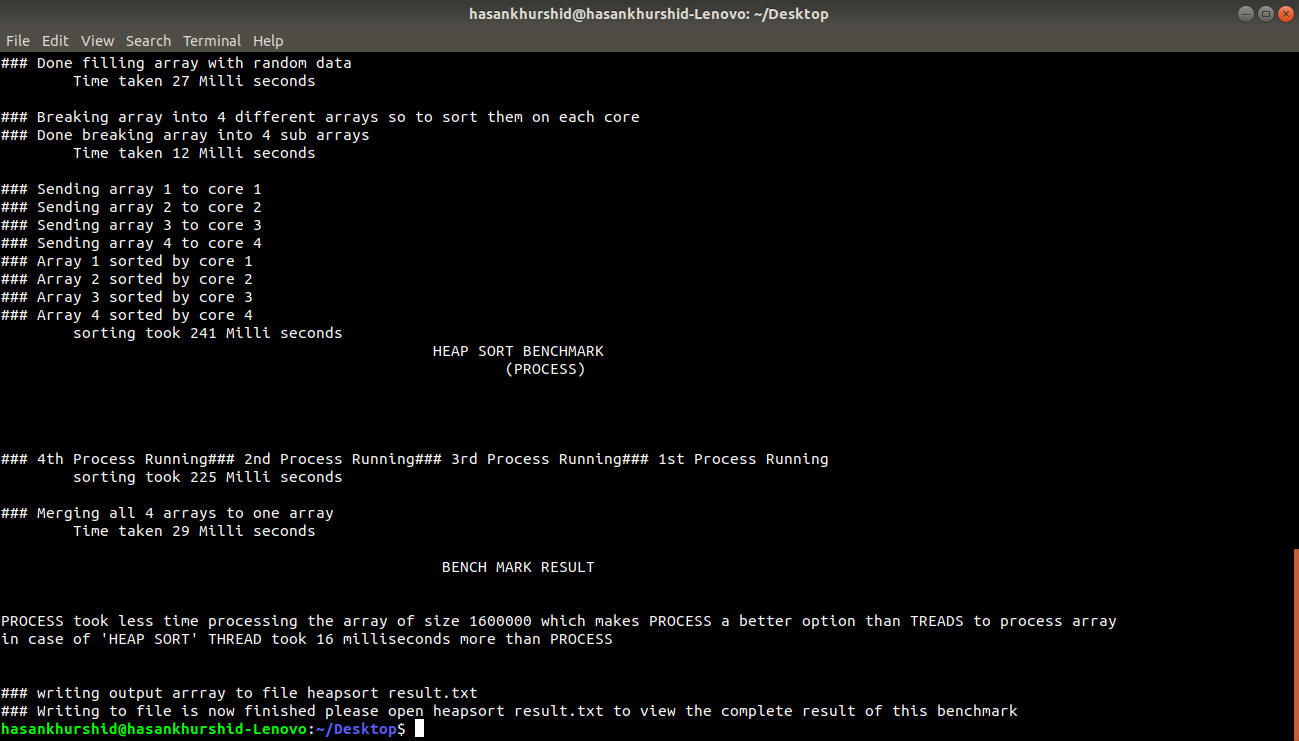
Radix Sort



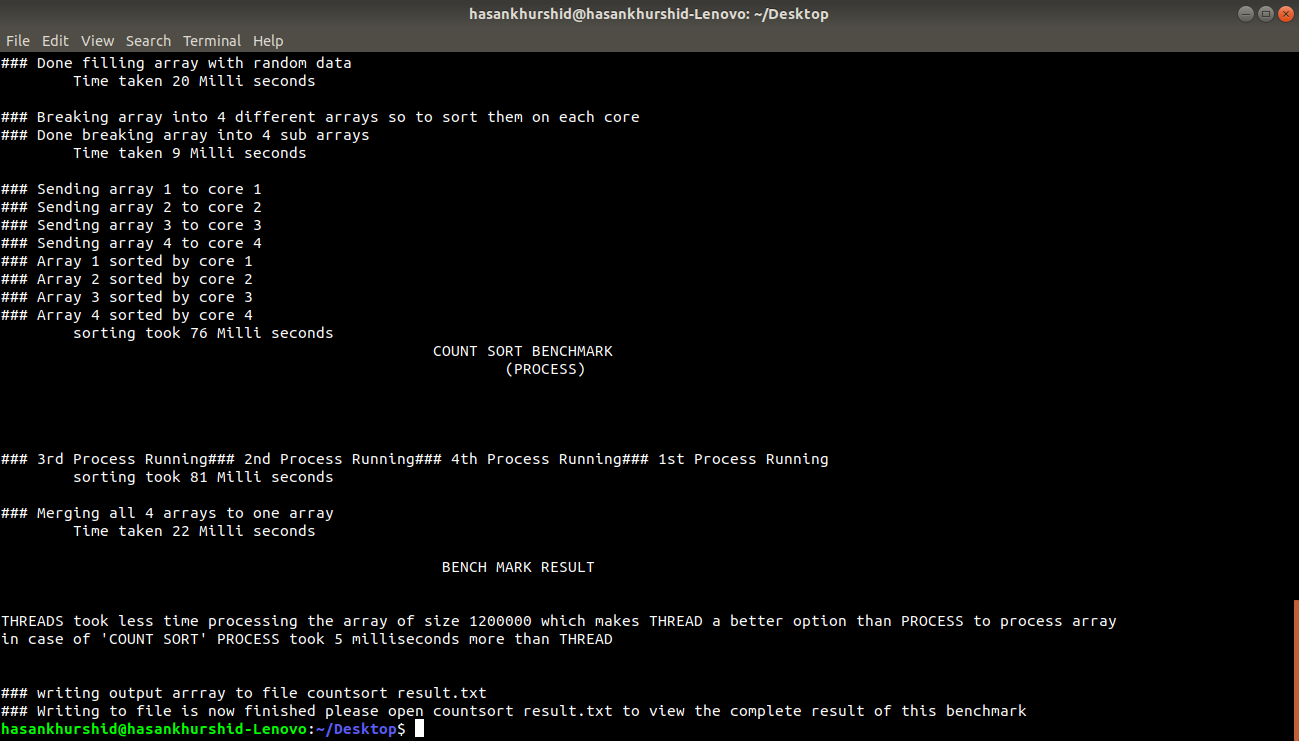
Insertion Sort



Heap Sort



Count Sort



Bubble Sort

