Faculty of Engineering and Applied Science
SOFE 3200U Systems Programming
Lab Report 5

Group Member 1
Name: Zachary Winn
Student ID: 100557505

Group Member 2
Name: Logan Macdonald
Student ID: 100579606

Group Member 3
Name: Justin Kaipada
Student ID:100590167

Group Member 4
Name: Georgy Zakharov
Student ID: 100588814

Date: 11/29/2017
Lab Section CRN: 44209

Questions

1. When working with Windows, what are the main structural differences you see that separate Windows and Linux?

Firstly and probably the biggest structural difference between Linux and Windows would be the fact that the user can access the source code while using Linux, but with Windows only a select few can access the source code. The Linux command line is also more useful than that in the Windows operating system, the Windows command line cannot nearly be used to the extent as the Linux command line, making Linux much more flexible. For example, with Linux it has the ability to stop at different run levels. With this, you can work from either the command line (run level 3) or the GUI (run level 5), however with Windows you can really only stay at the command line.

2. What are other types of sorts that exist? When compared to other methods, why is Bubble Sort not the most optimal?

Some other types of sorting algorithms include Merge Sort, Insertion Sort, Selection Sort, Heap Sort, Quick Sort and many others. When Bubble Sort() is called it has a best case running time of O(n) and a worst case running time of O(n^2), however the best case with Bubble Sort() only occurs when the set is already sorted before calling Bubble, so in this case we can ignore it's best case and focus on its worst case running time, which is a very inefficient running time. This is because with bubble sort it compares each element to it's previous and next element and performs only one switch at a time before moving on. This means that if there is at most one switch to make during its first iteration through the set, it then will have to run through the set for a second time, handling each element more than one time. A more efficient sorting algorithm for example would be the Merge Sort() algorithm which uses the divide and conquer method. This is a more efficient way to sort the algorithm because it allows for easier and smaller subsets to be more easily solved before merging them very quickly together to form the final sorted set. This algorithm unlike Bubble Sort() runs in O(n*log*n), which has a much quicker running time than Bubble Sort() for unsorted sets. Many other of the examples above share in having a more efficient running time than Bubble Sort() which is why it is not optimum.

Lab Tasks

The lab tasks have been submitted in separate folders with Makefiles for building them.

Task 1 is a simple batch script to run MS Visual Studio
Task 2 is the implementation of bubble sort in C

Group Participation

- **Justin**: Put together the lab report, tested tasks 1 and 2 and added a cmake file.
- **Logan:** Finished task 2**.**
- **Zach:** Answered lab task questions.
- **Georgy:** Finished task 1.