COP4530 ®C Data Structures, Algorithms and Generic Programming
Recitation 9
Date: 21st November 2011

Lab topic:

1 Assignment 5 discussion
2 Compiler Optimization Flags

1. Assignment 5 Discussion:

The major task in this assignment is:

- To choose and implement a hash function that takes a string as an argument and returns an integer. Your hash function should attempt minimize the number of collisions at least intuitively. Your hash function should attempt to address basic issues such as the following. If passed a same set of letters in different order, it should return different integer values (i.e. if hash("abcd") should not be equal to hash("bacd")).

- Design a data structure that does frequent searching in an efficient way. The data structure need not be something that you studied in class. You can think of something new, perhaps combining different data structures that we have studied. You can also make it efficient for commonly encountered words in English. While choosing a suitable data structures please think about its time complexity. Please understand a structure with search complexity O(n) is not a good idea. A search complexity of O(log n) may be acceptable. Try to come up with a structure that supports searching efficiently.

Once you have your data structure and a hash function you are to insert the words from /usr/share/dict/words file in four structures:

1. STL set
2. STL unordered_set (with the default hash function)
3. STL unordered_set (with your hash function)
4. The data structure you have chosen

For each word in the input file you have to print if it is found in the structures or not in the format specified in the assignment page.

In the end you have to print the following information for each of the structures mentioned above:

1. Time required for inserting the words in the structure.
2. The minimum time spent on a single search through that structure.
3. The maximum time spent on a single search through that structure.
4. The average time spent on a single search through that structure.

Please follow the output format specified in the assignment page. If you do not follow the naming convention or the output format you will have a lower grade. Also if you do not print the timing information your grade will be much lower and you will not be given extra time to print out these things. Your grade will depend on correctness and performance.

3. Compiler optimization flags:

Turning on optimization flags makes the compiler attempt to improve the performance at the expense of compilation time and possibly the ability to debug the program. The compiler performs optimization based on the knowledge it has of the program. Most optimizations are only enabled if an -O level is set on the command line. Otherwise they are disabled, even if individual optimization flags are specified. Use these flags as options in your GNU compiler and see the differences in execution time.

-O0: Reduce compilation time and make debugging produce the expected results. This is the default.

-O1: With -O, the compiler tries to reduce code size and execution time, without performing any optimizations that take a great deal of compilation time.

-O2: Optimize even more. GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff. As compared to -O, this option increases both compilation time and the performance of the generated code.

-O3: Optimize yet more. -O3 turns on all optimizations specified by -O2 and also turns on flags that perform optimizations like inlining, etc.

Other optimizations: You can take a look at the gcc/g++ documentation for other optimization flags. In particular, you can take a look at inter-procedural optimization and feedback optimization.

Note: The optimization flag might produce wrong results for code with bugs. So, please verify the correctness of your code with these flags.

References:

1 **Compiler Optimization Flags:** http://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html