Jonah Brooks
CS311 HW2
02-07-2012

1 UNIX Process Control

This is the write-up for assignment 2 of CS311. In this assignment, I created a program (h2.cpp) that accepts input from stdin, parses that input into case insensitive words delimeted by punctuation, sorts those words using a number of sort processes, then collects the sorted output and merges it together into one single list of unique words, returning this list via stdout. I also timed my code running 10^n words from the bible, and from random lipsum text. The graphs of the time taken for each test for values of n from 1 to 6 is on page 2 of this document.

2 Design Decisions

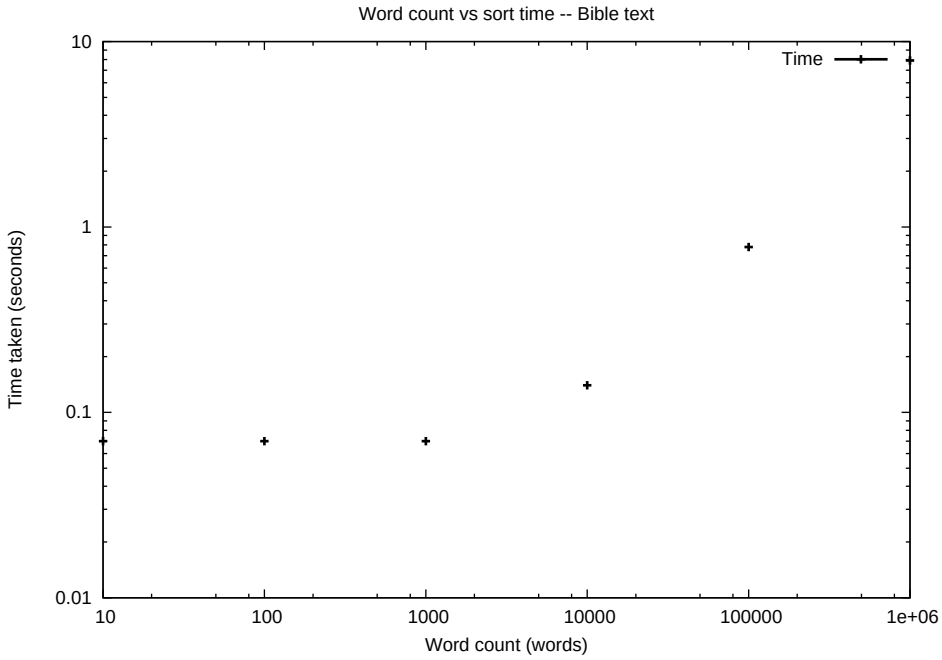
I chose to implement my code using a number of functions: parser, sorter, suppressor, normalize, and index_of_smallest. Each of these functions is designed for a single purpose; parser, sorter, and suppressor each hold the sections of code that should be run by the process of its titular type. This structure made it much easier for me to wrap my head around the parallel nature of this project, since I only needed to call a single function once I determined which process was running after the fork. This also allowed me to write my code in each function knowing that no other process type could access it.

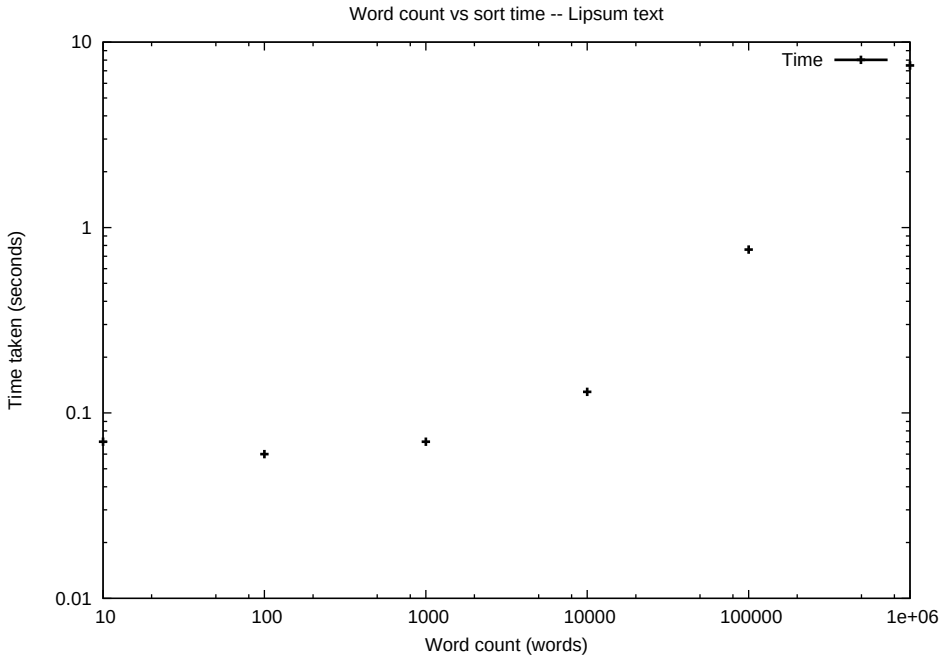
The other two functions are used for formatting and handling the input and output, respectively. normalize takes a word, forces every character to lower case, and removes all non-essential punctuation, leaving apostrophes and hyphens, but treating all other special characters as token delimeters. I chose these punctuations since I wanted words such as "that's" to count differently than "that," and because I wanted to count "compound-words" as one word. I left out all other punctuation because I did not want to count line of code such as "while(a=b+c)" to count as only one word. This has the unfortunate issue of breaking words on accented characters and non-standard apostrophes, but I think it works well, and efficently, for most documents an English speaker would want to parsing.

index_of_smallest was used to allow me to more easily merge the sorted lists back together into one single list. I chose to implement a very modified merge-sort algorithm for this purpose, in which I would take the smallest word from all visible words (effectively the ends of the pipes), print it to stdout (unless it is the same as the last printed word), then refresh the list of visible words before I repeat. This way I needed only one array that holds one word from each pipe, since I'm merging them into the output stream rather than a new array.

3 Commit Log

Commit Time	Commit Message
Tue, 7 Feb 2012 17:42:10 -0800	Finished my code and timings. Now I just need to finish my write-up, make sure it all runs on os-class, and tar it up.
Sun, 5 Feb 2012 15:27:30 -0800	Forgot to commit yesterday, but I got my code to the point where it was mostly functional with a few major bugs. Today I fixed those bugs and added the last features I want to add. I am mostly finished with my code now, except for some possible additional error checking and commenting.
Thu, 2 Feb 2012 17:18:21 -0800	All the pipes, forks, and execs are working correctly now. All that is left to do is suppress and output the sorted sub-lists. I also may want to process the words a bit before sending them to the sorters, such as removing punctuation and making them all lower case.
Thu, 2 Feb 2012 01:26:59 -0800	Almost done. A few final tweaks and it'll be ready. One annoying bug with forking still exists, but that's about it.
Wed, 1 Feb 2012 20:16:22 -0800	Long haul; created functions for each task I need to complete and placed in the pipe creation and forking code. I just have to close pipes, exec, and write the suppressor
Mon, 30 Jan 2012 17:53:24 -0800	Worked on my code a bit the other day just to whip together a quick parser. Refined it today to start making it do what the project actually needs. Still a long way to go


Word count vs sort time -- Bible text
Time
Time taken (seconds)
10
1
0.1
0.01
10
100
1000
10000
100000
1e+06
Word count (words)


Word count vs sort time -- Lipsum text
Time
Time taken (seconds)
10
1
0.1
0.01
10
100
1000
10000
100000
1e+06
Word count (words)