CO314: Systems and Network Programming Lab

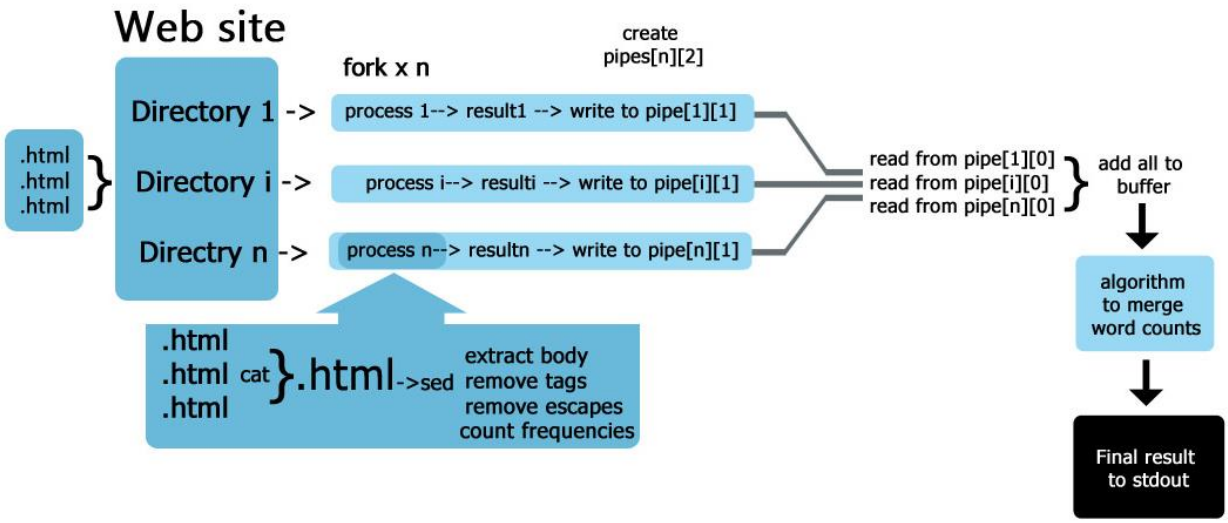
Project I - Iteration 3

Group 16 members

- **E/10/049**
- **E/10/170**

In this iteration we are going to process a large website with multiple directories containing HTML files. The program will run the HTML preprocessing and word count for the HTML files in each of the directories, concurrently. Once the word counts in all directories have been computed, they are merged to compile the overall word frequencies.

Our method



There are several directories in the web site and so many *.html files. First we merge all the .html files in one directory to single file. Then we created a separate process for that directory using fork. That process will do the entire html preprocessing to that file to extract necessary words and using the executable in iteration 1, it will generate word frequencies for that directory.

We created one pipe for each directory. Word frequency Results for each directory were written in to the write end of a pipe by each process. Parent wait until all the children finish writing and when they finished, parent reads all the pipes at the read end and merges all the results to a large buffer.

We used the hash table we implemented in iteration 1 to store and merge the results. We changed insertion function to the hash table so when similar word enters to the hash table, it will add the frequency values so there will be no duplicate values. Finally we print the data in the hash table to the standard output. Note that even we don't need a hash table to do that final merging, we used it because we implemented it in the iteration 1. So we can use it without creating a new method in this iteration.

IPC

We used pipes for communication between parent and children. But there is no communication between Childs. Each child writes the data directly to the separate pipes. So there is no conflict between Childs to write data. And since parent waits until all children finish their writing to pipes there is no conflict between Childs and parent.

Benchmark

We tested the program using 1 to 16 directories and 1 to 64 files in each directory. Obviously the time to process same number of files is decreasing when they are in multiple directories than in a single directory. because each directory has its own process. So we can say that processing is quick when we use multiprocessing.

		files in directory						
		1	2	4	8	16	32	64
directories	1	0.018	0.024	0.056	0.124	0.236	0.472	0.944
	2	0.024	0.052	0.114	0.232	0.476	0.936	1.876
	4	0.051	0.101	0.202	0.448	0.916	1.832	3.752
	8	0.098	0.188	0.426	0.902	1.796	3.702	7.56
	16	0.182	0.424	0.888	1.668	3.632	7.484	15.032

User time for processing @dualcore 2.00 Ghz processor. All values in seconds.