Word-Indexing

High Level Design

By CppBuzz.com, Jan 2015
last modified 07 Dec, 2018

1 Introduction

1.1 Problem

Create a multi-threaded text file indexing command line application in C++ that works as follows:

1. Accept as input a file path (e.g. /myfiles) on the command line

2. Have one thread that is responsible for searching the file path, including any sub-directories, for text files (ending in .txt)

3. When a text file is found, it should be handed off to a worker thread for processing, and the search thread should continue searching.

4. There should be a fixed number (N) of worker threads (say, N=3) that handle text file processing.

5. When a worker thread receives a text file to process, it opens the file and reads the contents one word at a time. Any character other than A-Z or 0-9 delimits words.

6. A master table in memory, shared between all threads, keeps track of all unique words

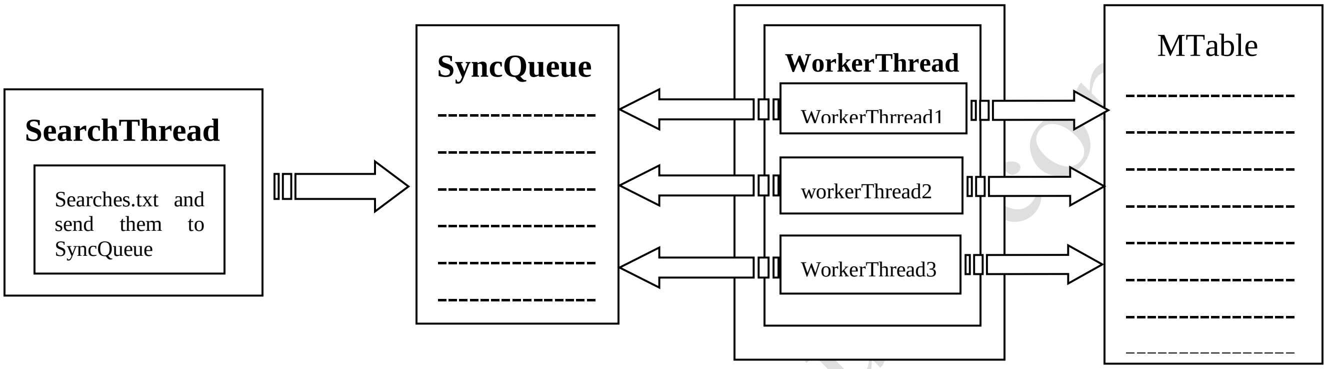
Encountered and the number of times it was encountered. Each time a word is encountered the count is incremented (or it is added to the table if not present). Words should be matched case-insensitive and without any punctuation.

7. Once the file search is complete and all text files finish processing, the program prints out the top 10 words and their counts.

We just want to find the top 10 words across a directory tree of text files.

2 Architecture

2.1 Architectural Diagram:



2.2 Modules

There are three modules SerachThread, SyncQueue and WorkerThread.

2.2.1 SearchThread

This module search for .txt file in the path specified as command line argument. In addition, it sends file to SyncQueue module. SearchThread stop working once searching is over.

2.2.2 SyncQueue

This module send the file in a synchronized Queue. This module provides file to WorkerThread module for processing. SyncQueue provides access of its Queue to only one WorkerThread at a time.

2.2.3 WorkerThread

This module has three workerthread and each thread get the file to process from SyncQueue module.

After getting the file, each workerthread reads the file and fetch words to save in a data structure called MTable. MTable contains unique words with there frequency.

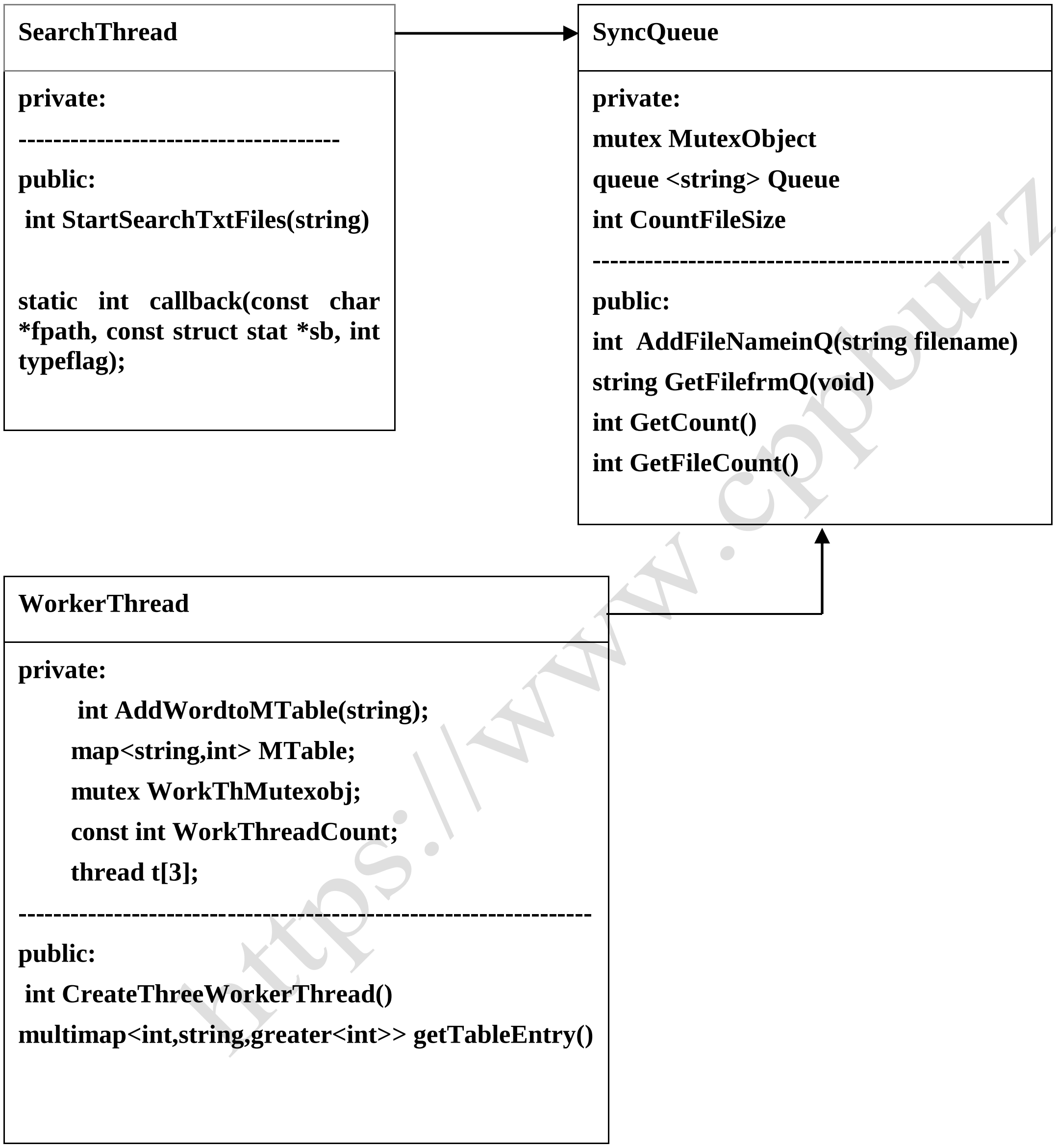
3 Class Diagram

This program has been divided into three classes:

1. SearchThread

2. SyncQueue

3. WrokerThread



SearchThread

private:

----

public:

int StartSearchTxtFiles(string)

static int callback(const char *fpath, const struct stat *sb, int typeflag);

SyncQueue

private:

mutex MutexObject

queue <string> Queue

int CountFileSize

--

public:

int AddFileNameinQ(string filename)

string GetFilefrmQ(void)

int GetCount()

int GetFileCount()

WorkerThread

private:

int AddWordtoMTable(string);

map<string,int> MTable;

mutex WorkThMutexobj;

const int WorkThreadCount;

thread t[3];

--

public:

int CreateThreeWorkerThread()

multimap<int,string,greater<int>> getTableEntry()

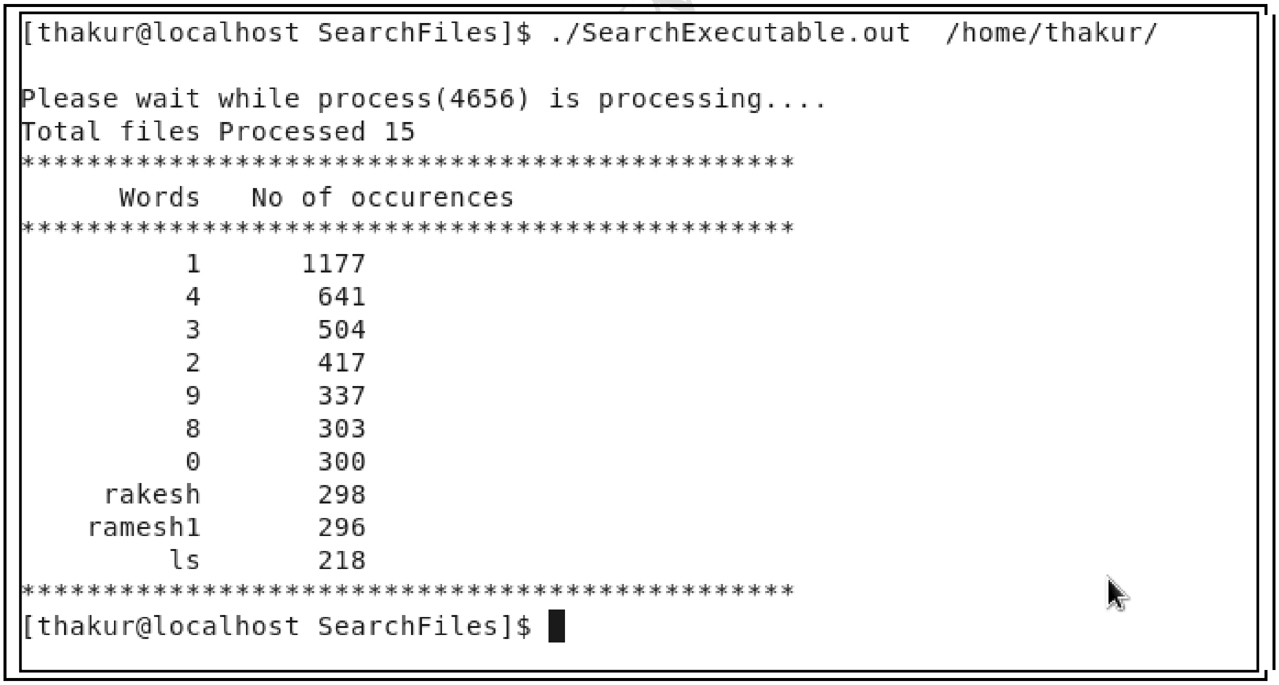
4 Development

Development in done on Fedora 12 using C++ 11 language.

4.1 Directory Structure:

```
SearchFiles→|
            | ---src
                |-- SearchThread.cpp
                |-- SearchThread.h
                |-- WorkerThread.cpp
                |-- WorkerThread.h
                |-- main.cpp
           | --- wordindex.out
           | --- Makefile
```

4.2 Output of Program:

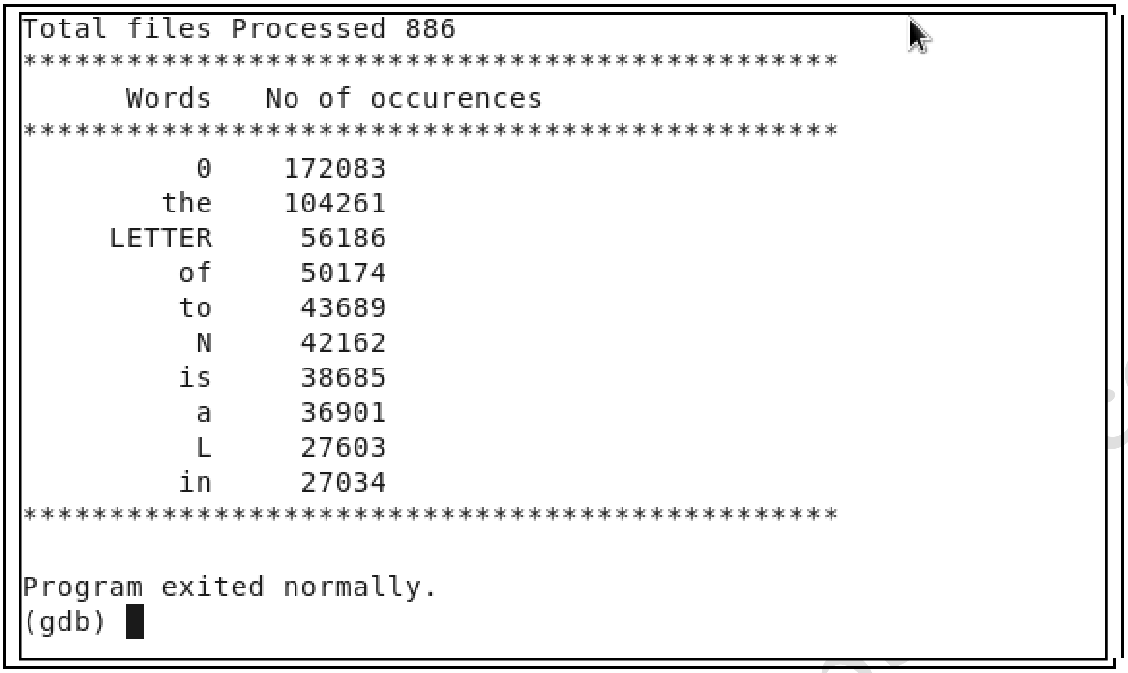


```
[thakur@localhost SearchFiles]$ ./SearchExecutable.out  /home/thakur/

Please wait while process(4656) is processing....
Total files Processed 15
***********************************************
      Words   No of occurences
***********************************************
          1      1177
          4       641
          3       504
          2       417
          9       337
          8       303
          0       300
     rakesh       298
    ramesh1       296
         ls       218
***********************************************
[thakur@localhost SearchFiles]$ █
```

4.3 Debugging

For debugging GDB is used.

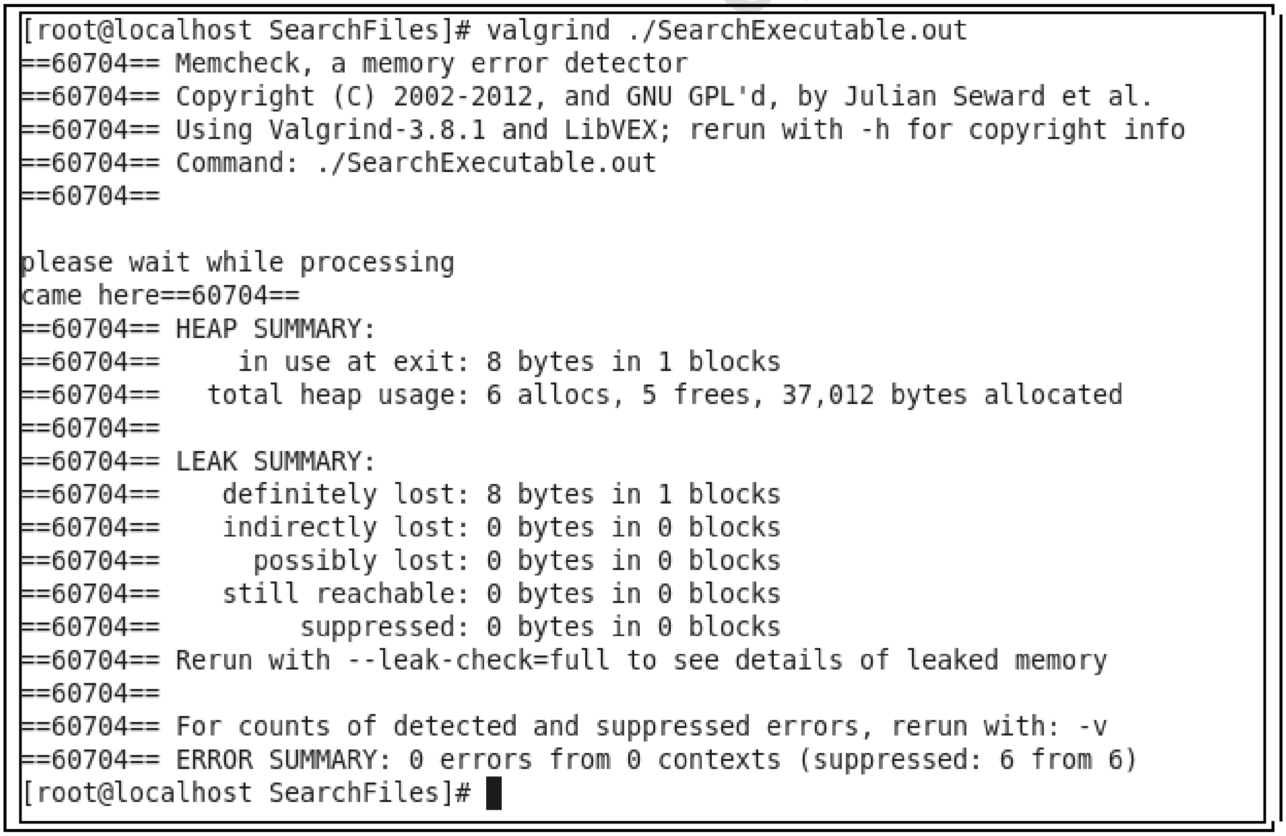


```
Total files Processed 886
***********************************************
      Words   No of occurences
***********************************************
          0    172083
        the    104261
     LETTER     56186
         of     50174
         to     43689
          N     42162
         is     38685
          a     36901
          L     27603
         in     27034
***********************************************


Program exited normally.
(gdb)
```

4.4 Memory Leaks

To find out memory Leak Valgrind tool is used.



```
[root@localhost SearchFiles]# valgrind ./SearchExecutable.out
==60704== Memcheck, a memory error detector
==60704== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
==60704== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
==60704== Command: ./SearchExecutable.out
==60704==

please wait while processing
came here==60704==
==60704== HEAP SUMMARY:
==60704==     in use at exit: 8 bytes in 1 blocks
==60704==   total heap usage: 6 allocs, 5 frees, 37,012 bytes allocated
==60704==
==60704== LEAK SUMMARY:
==60704==    definitely lost: 8 bytes in 1 blocks
==60704==    indirectly lost: 0 bytes in 0 blocks
==60704==      possibly lost: 0 bytes in 0 blocks
==60704==    still reachable: 0 bytes in 0 blocks
==60704==         suppressed: 0 bytes in 0 blocks
==60704== Rerun with --leak-check=full to see details of leaked memory
==60704==
==60704== For counts of detected and suppressed errors, rerun with: -v
==60704== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
[root@localhost SearchFiles]#
```

4.5 Known Issues

-Creating a multi map to sort the contents of map, which requires more memory, we can remove use of multi map.

-WorkerThread module returns the multi map, to print this multi map in main function I am creating one extra multi map to save multi map returned by WorkerThread module.

4.6 Glossary

-MTable is a data structure, which contains words with their frequency.

-WorkerThread1, WorkerThread2 and WorkerThread3 are three-worker thread, which are part of WorkThread and responsible for filling words in MTable.

-Queue is synchronized queue, which contains file.

5 Test Cases

S.No	**Test Case**	**Pass/Fail**	**Expected Result**	**Actual Result**
1	Input a directory which Is blank (no .txt file)	Pass	Total File Processed is 0	Total File Processed is 0
2	Input a directory which Has a single .txt file but no words in it	Pass	Total File Processed is 1 But list has 0 words	Total File Processed is 1 But list has 0 words
3	Input a directory which Has single .txt	Pass	Total File Processed is 1 And will show list of words With occurrence	Total File Processed is 1 And will show list of words With occurrences
4	Input a directory which Has two .txt file	Pass	Total File Processed is 2 And will show list of words With occurrence	Total File Processed is 2 And will show list of words With occurrence
5	Input a directory which Has three .txt file	Pass	Total File Processed is 3 And will show list of words With occurrence	Total File Processed is 3 And will show list of words With occurrence
6	Input a directory which Has four .txt files	Pass	Total File Processed is 4 And will show list of words With occurrence	Total File Processed is 4 And will show list of words With occurrence
7	Input a directory which As five .txt files	Pass	Total File Processed is 5 And will show list of words With occurrence	Total File Processed is 5 And will show list of words With occurrence
8	Input a directory which Has seven .txt files	Pass	Total File Processed is 7 And will show list of words With occurrence	Total File Processed is 7 And will show list of words With occurrence
9	Input a drectory which Has nine .txt files	Pass	Total File Processed is 9 And will show list of words With occurrence	Total File Processed is 9 And will show list of words With occurrence
10	Input a drectory which Has ten .txt files	Pass	Total File Processed is 10 And will show list of words With occurrence	Total File Processed is 10 And will show list of words With occurrence
11	Input a directory with Twenty .txt files	Pass	Total File Processed is 20 And will show list of words With occurrence	Total File Processed is 20 And will show list of words With occurrence
12	Input a directory which As . (dot) only (./SearchExecutable .)	Pass	It t should process all text Files of current dir	It processed all text files Of current dir
13	Input a invalid dir (./SearchExecutable n)	Pass	Error msg : Directory doesn't exist	Error msg : Directory doesn't exist
14	Input a root directory (./SarchExecutable /)	Pass	It should all text file exist In the computer	It processed 886 text file for me (on my computer)
15	Input a root directory (./SarchExecutable /) And run with gdb	Pass	Program exited normally	Program exited normally
16	Memory Leak detection Using Valgrind tool	Pass	There should not be memory Leak more than Bytes	This program has 8 Bytes Of memory leak only

If you want support for any other project then drop email to Admin@cppbuzz.com

Thank you,
Admin
CppBuzz.com, Chicago USA