Programming Assignment #2 Documentation Savishwa Gaur

Sxg180113

**Filecnt:**

To compile the program use “g++ -std=c++17 filecnt.cpp -o filecnt.exe”, which specifies c++17 standard in order to make use of the filesystems library and sets the executable file to “filecnt.exe”. The executable was created on my local machine.

In order to run the program use “./filecnt.cpp <directory name>”, which runs the filecnt executable file and passes in the directory of interest.

The program contains 3 static int global variables fileCount, directoryCount, and finalSize all defaulted to 0 which represent the number of files in the passed directory, the number of directories in the passed directory, and the size in bytes of all the files in the passed directory respectively.

The program contains the following functions:

* **Int main (int argc, char \*\*argv):** This is the main function where the directory is stored in a variable called dir. The directory is extracted by using argv[1]. A path object called filePath is then created with the dir string passed through. The filecnt function (described below) is then evaluated. If the filecnt function returns true, the total number of directories in the passed dir, total number of files in the passed dir, and the total number of bytes occupies by all files in the passes dir are displayed. If the filecnt function returns false, the user is informs that the passed dir cannot be found.
* **Bool filecnt (const fs::path& filePath, int directoryNest):** This function receives the filePath path object created in the main function and a directoryNest parameter which controls the position of the iterator in the passed directory. A validFlag of type bool is declared as false by default, this flag indicates if the filePath is exists. A check is made using the filesystem::exists(path) and filesystem::is\_directory(path) functions to ensure that the passed filePath is valid. If filePath passes the test, then validFlag is changed to true. Then, a for loop is used with filesystem::directory\_iterator(path) to explore the passed directory. A check is made on the status of the current item the iterator is pointing to. If the item is a directory (using filesystem::is\_directory()), then the directoryCount is incremented and the filecnt() function is called again with the passed path object being the current item and the directoryNest being directoryNest + 1. If the item is a file (using filesystem::is\_regular\_file()) fileCount is incremented and the size of the current item (filesystem::file\_size()) is added to finalSize. The function returns validFlag back to the main function to determine what to display to the user.

Sample Input/Output:

* Valid Input:
  + Input: ./filecnt.exe testDir
  + Output: The total number of directories in directory testDir is: #

The total number of files in directory testDir is: #

The total number of bytes occupied by all files in directory testDir is: #

* + Where ‘#’ is the correct number of directors/files/bytes
* Invalid Input:
  + Input: ./filecnt.exe invalidDir
  + Output: The following directory cannot be found: invalidDir

**Filedisp:**

To compile use “g++ filedisp.cpp -o filedisp.exe” to set the executable file to filedisp.exe, which was created on my local machine

In order to run the program, use pipelining: “<command that has standard output> | ./filedisp.exe <option>”. This runs the executable of filedisp and passes both the standard output of the specified command and a display option that can be either “-f”, “-d”, or “-b”.

The options correspond to the following outputs:

* **-f:** Displays the lines of the input which contain the word “files:
* **-d:** Displays the lines of the input which contain the word “directory”
* **-b:** Displays the lines of the input which contain the word “bytes”

The program contains the following functions:

* **int main (int argc, char \*\*argv):** This is the main function where the option and input are rad in. The option is read into the constant string variable option using argv[1]. The input is stored in a map with int values for the key and string values for the value. The int key value corresponds to the line number of the input and the string values corresponds to the actual contents of the corresponding line number. An int variable called index is decalared and defaulted to 0 to represent the starting line position, and a string variable called lineInput is declared to store the current line of input. A while loop with the condition getline(cin, lineInput), which loops until cin cannot read anymore lines from the input, is then ran. Inside the loop each line of input is concatenated with the newline character and fullInput[index] is assigned the concatenated lineInput value. The index is then incremented. The eval() function (explained below) is then evaluated. If it returns false, the user is informed that the about options could not find anything in the input string.
* **bool eval (string option, map<int, string> fullInput):** This function accepts a string value for the option, and a map<int, string> fullInput which contains the passed line by line input. A flag called validFlag of type bool is declared and defaulted to false. This represents if the passed option was able to find anything in the input string. Then a for loop is used to traverse the fullInput map. A check is then made using option.compare() to determine the correct display option. If the passed option is not one of the three listed above, the user is informed that the option is invalid. If the option is valid, a check is then made again for the keyword of the passed option using fullInput[i].find(). If the keyword is found, fullInput[i] is sent to standard output and the validFlage is set to true. This function returns the value of validFlag to the main function to determine wheter to display the input error message.

Sample Input/Output for this program is shown in the next page as a part of the sample input/output of the two programs pipelined together. This is because this program is meant to be pipelined with the filecnt program.

**Pipelined filecnt and filedisp:**

To run use “./filecnt.exe <directory name> | ./filedisp.exe <option>”. This runs the filecnt executable and send the output to the filedisp executable which also accepts a display option.

The file “PA02.scriptout” contains the testing of these programs.

Sample Input/Output:

* Valid Input:
  + Input ./filecnt.exe testDir | ./filedisp.exe -f
  + Output: The total number of files in directory testDir is: #
  + Input: ./filecnt.exe testDir | ./filedisp.exe -d
  + Output: The total number of directories in directory testDir is: #
  + Input: ./filecnt.exe testDir | ./filedisp.exe -b
  + Output: The total number of bytes occupies by all files in directory testDir is: #
* Invalid Input:
  + Input: ./filecnt.exe invalidDir | ./filedisp.exe <valid option>
  + Output: Nothing found in input string
  + Input: ./filecnt.exe <valid director> | ./filedisp.exe <invalid option>
  + Output: Invalid option: <invalid option>

Nothing found in input string

**Important Note:**

While trying to run these programs on the UTD linux server, I was unable to use the command “g++ -std=c++17 filecnt.cpp -o filecnt.exe” as I was getting the error “g++: error: unrecognized command line option ‘-std=c++17’”. I also tried to use sftp to transfer my executables to the UTD linux server and run those but I got the error message: “-bash: ./filecnt.exe: cannot execute binary file”. I tried to execute the commands in the following discussion post:

Graphical user interface, text, application, Teams

Description automatically generated

However, I was unable to execute the sudo commands. Here is the output of the sudo commads:

Text

Description automatically generated

This similar image is sent to standard output with sudo command and the installation does not work. I tried to ssh into the UTD linux server from both my local Mac machine as well as a VM from virtual box and got the same result when running the sudo commands.

For this reason, the executables for programs and the “PA02.scriptout” file were created on my local Mac machine.