Programming Assignment 3:

FileCnt/FileDisp

Description: FileCnt is a program that accepts an absolute path from the user and recursively explores the path by opening directories within directories. While traversing the path given the program counts the number of directories open, the number of plain files (not directories), and the number of bytes contained in those plain files. Continuing, the program FileDisp was serves its purpose to demonstrate pipelining. The program takes in input from FileCnt and displays the corresponding line depending on the option given. Option d displays directory count, option f displays the file count, and option b displays the byte count. I was able to implement the programs so that they work well in unison.

Prerequisites (how to use it): The program was designed to run on a computer utilizing the UNIX/LINUX operating system. Therefore, you need either a computer with UNIX/LINUX installed or a virtual box running UNIX/LINUX on your Windows system. Next you need to open your terminal on your system and run the command “make -f makefile.PA3” this will compile the c++ scripts. Once completed you can now enter “./filecnt> <path> | ./filedisp -<option>” and watch the program do its job.

How it works: FileCnt accepts whatever command it was given as the path and starts by making sure the user has access to the directory. Once it has access the program checks each individual files inode which signifies what type of file it is. The program will then increment the necessary count variables as it traverses the given path. Once the program is done traversing the print function will be called which displays the results in the format given. FileDisp accepts the given output and checks for the word and option to know which line to display. The source code with a more detailed look into the workings of the program is shown below. The comments explain what each section of code does for a better understanding. The text files used for testing are included in the folder titled Text Documents.

#**include** <iostream>

#**include** <fcntl.h>

#**include** <sys/stat.h>

#**include** <sys/types.h>

#**include** <unistd.h>

#**include**<stdio.h>

#**include**<dirent.h>

#**include**<string.h>

#**include** <fcntl.h>

**using** **namespace** std;

**class** FileCnt {

**public**: // class that holds the counts for the monitered variables

**int** dirCount; // initializes the variables

**int** fileCount;

**int** byteCount;

FileCnt() { // default constructor

dirCount = 0;

fileCount =0;

byteCount = 0;

}

**void** **traverse**(string str) { // method that recursively traverses the given directory

**const** **char** \*c = str.c\_str(); //capture the file path

**if** (access(c, R\_OK) == 0) { // checks if the user has read access skips otherwise

DIR \*pwd = opendir(c); // opens the directory

**struct** stat buf;

**struct** dirent \*dirRead;

**if** (pwd != nullptr) { // makes sure the directory is not a null pointer

**while** ((dirRead = readdir(pwd)) != nullptr) { // reads everything in the directory

stat(dirRead->d\_name, &buf);

**if** (strcmp(dirRead->d\_name, ".") != 0 && strcmp(dirRead->d\_name, "..") != 0) {

**if** (dirRead->d\_type == DT\_DIR) { // if directory

dirCount++; //increment count

**char** tmp[4096]; // buffer one to make the new path

**char** tmpTwo[4096]; // buffer two to make the new path

string newDir = strcpy(tmp, c);

string dirName = strcpy(tmpTwo,dirRead->d\_name);

newDir += "/" + dirName; // constructs the new file path

traverse(newDir); // recursively traverses the next directory

} **else** **if** (dirRead->d\_type == DT\_REG) { //if plain file

fileCount++; // increment count

byteCount += buf.st\_size; // update the byte size

}

}

}

}

}

}

**void** **print**(string n) { // creates print statements and writes them to standard outptut

string lineOne = "The total number of directories in directory " + n + " is: " + to\_string(dirCount);

string lineTwo = "The total number of files in directory " + n + " is: " + to\_string(fileCount);

string lineThree = "The total number of bytes occupied by all files in directory " + n + " is: " + to\_string(byteCount);

printf("%s \n", lineOne.c\_str());

printf("%s \n", lineTwo.c\_str());

printf("%s \n", lineThree.c\_str());

}

};

**int** **main**(**int** argc, **char** \*argv[]) {

**char** tmp[4096];

strcpy(tmp, argv[1]); // captures the given file path

FileCnt counter; // creates an instance of the object

counter.traverse(tmp);

counter.print(tmp);

**return** 0;

}

#**include** <iostream>

#**include** <fcntl.h>

#**include** <sys/stat.h>

#**include** <sys/types.h>

#**include** <unistd.h>

#**include**<stdio.h>

#**include**<dirent.h>

#**include**<string.h>

#**include** <fcntl.h>

**using** **namespace** std;

**int** **main**(**int** argc, **char** \*argv[]) {

string outputOne; // creates a string for the next lines

string outputTwo;

string outputThree;

string d = "directories"; // strings for comparisons

string f = "files";

string b = "bytes";

getline(cin, outputOne); // reads standard input

getline(cin, outputTwo);

getline(cin, outputThree);

string arg = argv[1]; // captures the option

**if** ((arg == "-d") && (outputOne.find(d)) != string::npos) {

cout << outputOne << endl;

} **else** **if** ((arg == "-f") && (outputTwo.find(f)) != string::npos) {

cout << outputTwo << endl;

} **else** **if** ((arg == "-b") && (outputThree.find(b)) != string::npos) {

cout << outputThree << endl;

} **else** {

cout << "Error: Incorrect input" << endl;

}

**return** 0;

}