Virginia Tech
Bradley Department of Electrical and Computer Engineering
ECE-3574: Applied Software Design Fall 2014

Homework 3

Submission Details

You must submit the solutions for this homework as an electronic submissions using Scholar (under ECE3574 → Assignments → Homework 3). The submission must be a gzipped tar file (.tar.gz) with your source code. Include all necessary project files, but no binary or compiled files. Your program will be run to evaluate its correctness, and the source code will be reviewed for adherence to the Qt programming style. Your program must run on Ubuntu 14.04.1 and compile/build using the GNU C/C++ compiler and the qmake/make tools. The following information must be included at the top of each of your source files as comments: your full name, your student ID number, your email address, class (ECE 3574), and the title of the assignment (Homework 3). The submitted file must be given a name in the following form: ***LAST_FIRST_hw3.tar.gz*** where LAST is your last or family name and FIRST is your first or given name. Paper, email or Drop Box submissions will not be accepted. All work must be submitted by the announced due date/time. **Late submissions will not be accepted!** (Don't do it! You have been warned!)

Questions

Use the Homework 3 forum in the Discussion Board area of the class web site to ask questions about this assignment. Do not post questions that contain specific information about the solution.

Honor Code

As stated in the syllabus, in working on homework and projects, discussion and cooperative learning are allowed. However, copying or otherwise using another person's detailed solutions to assigned problems is an honor code violation. See syllabus for details.

Learning Objectives

The primary learning objectives of this assignment include learning to perform command-line processing of arguments and creating and linking libraries. Additional learning objectives include the correct use of iterators and container classes.

Utility Description

diskusage reports, in various formats, the amount of disk space used in a directory tree. It traverses all files and computes total file sizes, providing different levels of detail (depending on command line arguments) in its report.

The command-line format of invoking diskusage is as follows:

./diskusage -depth=(all|<uint>) (-b|-k|-m) -<type> resource0 [resource1 […]]

Where “resource” is used to mean a file, directory or symbolic link. At least one resource, resource0, must be provided to the utility and further resources are optional. Whenever an invalid or an otherwise undefined situation arises the program must exit with the “help” description of the utility.

Options for diskusage

All functionality used to parse the command-line options must be built to form a library. You will use at least one header file to **declare** non-static **definitions** that appear in your library but the executable must link to the library and not the *.o object file where argument parsing **defintions** appear. You can verify this is the case with the arguments to g++ during compilation and linking. You may use libs2.tar.gz under Resources in Scholar to parse arguments. You may also extend the functionality provided in that library.

Depth Argument:

-depth=all	Perform no filtering based on depth.
-depth=<uint>	Show no resources more than <uint> levels deep. This argument may not be negative, must be an integer and may be multiple digits long.

Note that “-depth=0” shows no more than the resource arguments provided to diskusage, “-depth=1” shows no more than 1 level deeper than “-depth=0” and so on.

Unit Argument:

-b	Sizes are shown in bytes with a “B” suffix.
-k	Sizes are shown rounded up to the nearest kilobyte with a “kB” suffix.
-m	Sizes are shown rounded up to the nearest megabyte with a “MB” suffix.

Note that these flags do not affect how the sizes are calculated. They affect only how the sizes as they are displayed. This is referred to as the rounded size.

Type Argument:

f Allow files to be shown. Does not affect symlinks.

d Allow directories to be shown. Does not affect symlinks.

s Allow symbolic links to be shown. Regardless of what the symlink points to.

A valid type argument is “-fds” with up to 2 of the character flags absent. That is, the type argument may be “-fds”, “-fd”, “-fs”, “-f”, “-ds”, “-d” and “-s” but **not** “-“ or any other string. If “f” is absent no files may be shown, if “d” is absent no directories may be shown and if “s” is absent no symbolic links may be shown.

Behavior

Visit and Print Order:

Given a list of resources you should visit resources at the same level in lexical order and, for directories, visit their children in pre-order (i.e. A/ and B/ should always print A/, A/’s children, B/, then B/’s children regardless of the way they’re specified on the command line because A comes before B in lexical order). Symbolic links should not have their children traversed. The output should be printed in this order as well.

Filtering:

Both the depth and type arguments may restrict which resources are printed. When checking which resources should be printed it must pass at least one of the choices for type (i.e. file OR directory OR symlink for “-fds”) AND must pass the depth test.

Size Calculation:

A file’s total size and direct size are the size of its contents in bytes. A symlink’s total size and direct size are the size of its targets direct size. A directory’s total size is its direct size plus the total size of its contained items and its direct size is reported by QFileInfo::size. Note that these sizes are not affected by the unit argument.

Printing:

The order in which resources are printed is specified in “Visit and Print Order” when the program determines a resource should be printed it should print the rounded size in the left-hand column and the path to the resource in the right-hand column.

Notes

Don’t forget that even empty folders have a size that needs to be taken into consideration when calculating the total. This likely varies with your operating system/disk format.

Check that subdirectories in the traversal each have their own totals properly calculated. Compare your results to the *nix command **du**.

Citation

This homework is based on Section 8.2.3 Exercise 1 from Design Patterns in C++ with Qt 4, 1st Edition.