Development test

Background

When hiring new people to work for ProFamily we wish to be able to early assess the capabilities regarding design and coding and have decided to design a relatively small task that suits to demonstrate this.

C++ Development task for potential new programmers

Description

The ProFamily products were previously stored in Microsoft Visual SourceSafe (VSS). VSS supports linking the same files to different projects making the file appear in several places in a directory structure. Linking files in that way is not supported in any modern revision control systems. When we migrated to subversion late last year, we needed to eliminate that use. Using VSS to do this seemed inefficient, work intensive and slow, and would at least require us to write some VSS-specific code to solve that task. We therefore chose to write a tool that would find those links by identifying similar files directly in the file system. If you have been asked to perform the “Basic” developer test, unique files are identified by file name alone. If you have been ask to perform the “Extended” developer test, the contents of the file will also have to be matched.

Specification

General

The project should contain a readme.txt-file with a brief explanation of the functionality as well as limitations and considerations.

The program may be coded on Windows or Linux/Unix as preferred by the developer.

x64-compilation is desired, although not required.

ANSI Standard C++ is required.

The program should not be multithreaded, but a line or two about how it could be altered to make use of multithreading may be placed in the readme.txt-file.

STL usage is allowed and desired, keeping in mind the readability clause below.

Usage of object orientation is preferred.

Performance is an issue; the program should be fast even with thousands of files and projects.

There should be no practical limit on number of files and projects as well as folder depth given huge projects and the resources normally available on a standard modern workstation. Comments regarding this may be placed in readme.txt.

Readability is important. You should not need a doctor’s degree in C++ to read the code.

Open source code may be allowed for small, isolated tasks where no standard functions cover the need. Such functions must be isolated behind interfaces or encapsulated in functions to allow for easy replacement. These kind of modules should be bundled with the project and the dependency should be noted in the readme.txt-file.

The code should be written in a portable manner. Where no standard C/C++ functions exists, interfaces or functions should be used to minimize the effort to port to other platforms.

Specifics

The program should take at least one parameter on the command line, specifying the path to the base folder of the tree of files to scan.

The tree should be scanned for source files of at least the types *.c, *.h, *.cpp, and *.rc.

Files residing in exactly the same projects should be presented together by listing the full path to all the projects they exist in first, and listing the individual files indented by one tab stop.

Any additional detail about projects and files should be placed after the file name.

Example:
We have tree projects, A,B and C
We have three files, X,Y,Z.
Files X and Y exist in projects A and B.
File Z exist in all projects.
The output should be as follows:

```
/A
/B
        X
        Y
/A
/B
/C
        Z
```

It should be possible to view the output on the screen as well as storing it to a file.

References

The CM&L coding and style guide should be followed to the extent practical to this task given the time and size of it. Comments about the guide itself, why parts of it has not been considered should be placed in the readme.txt.

For candidates who has not got access to the CM&L coding and style guide, the code should be

- Simple and readable
- Commented where comments add value and makes sense, i.e. comment complex code, not the obvious
- Object oriented
- Without leaks, dangling pointers or handles etc.

- Using C++ standards, avoiding platform specific functionality
- Use of C++ code and libraries should be preferred over their C counterparties