User's Guide

CodeSuite-LT Version 1.5

a product of

S.A.F.E.

Software Analysis & Forensic Engineering Corporation

Contents

Introduction

CodeSuite-LT

CodeSuite-LT® is a "lite" version of the CodeSuite® collection of computer code analysis tools. The individual tools that comprise the suite of tools include BitMatch®, CodeCross®, CodeDiff®, CodeMatch®, FileCount™, FileIdentify™, FileIsolate™, and SourceDetective®, all of which are described below.

BitMatch

BitMatch uses fast, simple algorithms to compare thousands of executable binary files in multiple directories and subdirectories to thousands of other executable binary files or source code files in order to determine which files are the most highly correlated. BitMatch is particularly useful for finding programs that have been copied, but where you only have access to the program executable binary files and not the source code.

BitMatch compares every file in one directory with every file in another directory, including all subdirectories if requested. BitMatch produces an HTML basic report that lists the most highly correlated pairs of files. You can click on any particular pair listed in the HTML basic report see an HTML detailed report that shows the specific items in the files (strings or identifiers) that caused the high correlation.

BitMatch examines all text strings, comments, and identifier names that it can find in the executable files in order to determine copying. If a specific user message or a unique subroutine name is found in two files, there is a possibility that one was copied from the other. Note that BitMatch gives only a rough determination whether copying took place. False positives and false negatives are both possible. CodeMatch is needed to compare source code to help make a definitive determination.

CodeCross

CodeCross uses a fast, simple algorithm to compare thousands of source code files in multiple directories and subdirectories to find programming statements in one file that have been commented out of another file -- a possible sign of copying.

CodeCross compares every file in one directory with every file in another directory, including all subdirectories if requested. CodeCross produces an HTML basic report that lists all files in one folder that have statements that match comments in files in the other folder. You can click on any particular file pair listed in the HTML basic report see an HTML detailed report that shows the specific lines in the files that match.

CodeDiff

CodeDiff uses a fast, simple algorithm to compare thousands of source code files in multiple directories and subdirectories to find files that are exact matches or nearly exact matches. CodeDiff looks for identical lines in pairs of source code files. While not as sophisticated or as accurate as CodeMatch, CodeDiff runs much faster. CodeDiff is particularly useful for comparing files where it is already known that many of the files are nearly identical. CodeDiff can be run as a precursor to running CodeMatch when attempting to find source code plagiarism.

CodeDiff compares every file in one directory with every file in another directory, including all subdirectories if requested. CodeDiff produces an HTML basic report that lists the most similar pairs of files based on matching lines of source code in the files. You can click on any particular pair listed in the HTML basic report see an HTML detailed report that shows the specific lines in the files that are different.

CodeMatch

CodeMatch compares thousands of source code files in multiple directories and subdirectories to determine which files are the most highly correlated. This can be used to significantly speed up the work of finding source code plagiarism, because it can direct the examiner to look closely at a small amount of code in a handful of files rather than thousands of combinations. CodeMatch is also useful for finding open source code within proprietary code, determining common authorship of two different programs, and discovering common, standard algorithms within different programs.

CodeMatch compares every file in one directory with every file in another directory, including all subdirectories if requested. CodeMatch produces an HTML basic report that lists the most highly correlated pairs of files. You can click on any particular pair listed in the HTML basic report see an HTML detailed report that shows the specific items in the files (statements, comments, strings, identifiers, or instruction sequences) that caused the high correlation.

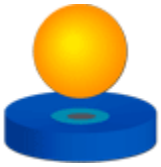
CodeMatch uses unique algorithms to find various different ways that source code files are correlated. These algorithms can find directly copied source code and even source code that has been modified to avoid detection.

FileCount

FileCount is a utility that counts the number of files, non-blank lines, and bytes in a large set of files in a directory tree. FileCount is useful when using CodeDiff to generate statistics about a set of source code files.

FileIdentify

FileIdentify is a utility that examines all of the file types in a given directory, or an entire directory tree, and reports the associated programming languages if known.



FileIsolate

FileIsolate is a utility that allows files to be selectively deleted from a large group of files in an entire directory or directory tree. FileIsolate is useful when examining a large number of files but only certain files are of interest and all other files can be deleted to make searches faster.

Copyrights, Trademarks, Patents

Copyrights

Trademarks

Patents

CodeSuite-LT is covered by U.S. patents 7,503,035, 7,823,127, 8,255,885, 8,261,237, and the following pending U.S. patents: 111/467,155, 12/253,249, 12/870,817, and 12/871,808.

Using CodeSuite-LT

System Requirements

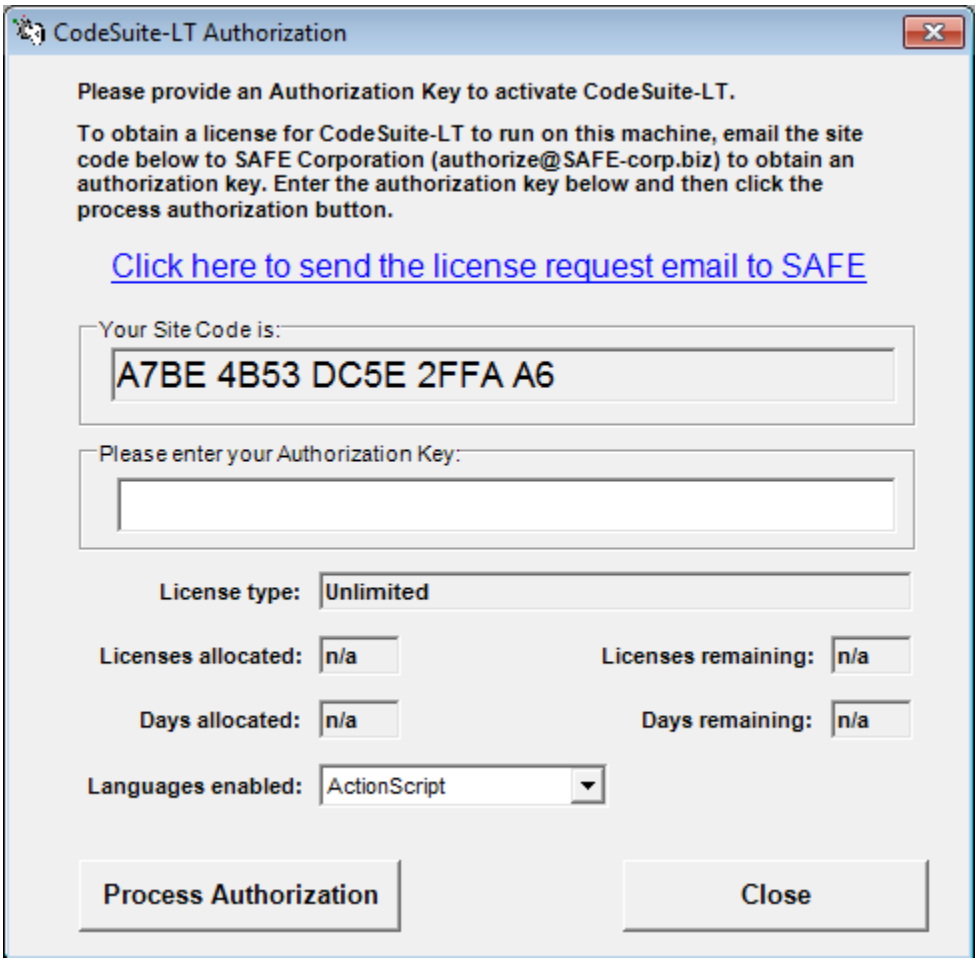
CodeSuite-LT will run on any computer using any of the following versions of the Microsoft Windows operating system:

- Windows Vista
- Windows 7
- Windows 8
- Windows 10

Note that CodeSuite-LT will not run on a virtual system and may not run on some systems using a remote desktop.

Licenses
Licenses must be purchased from SAFE Corporation. The FileCount, FileIdentify, and FileIsolate functions of CodeSuite-LT do not require a license.

To request licenses, open the authorization form shown below from the Help menu. Send the site code to SAFE Corporation and the number of licenses requested, along with appropriate payment. SAFE Corporation will send back an Authorization Key that must be entered into the field in the form. Press the process authorization button and the form will show the following information. Licenses are enabled for only one PC and cannot be transferred to another PC.



License Type

The license can be one of three types.

- **File size based.** Used to examine a fixed amount of bytes of source code. Licenses are used up as source code is examined.
- **Time based.** Used to examine any amount of code for a fixed number of days.

- **Unlimited.** There is no limit on the number of megabytes that can be examined and there is no expiration date.

Licenses Allocated and Licenses Remaining

These fields indicate the number of licenses that were originally allocated and how many unused licenses remain. These fields are valid only for a megabyte based license. For other licenses, the fields are not applicable ("n/a").

Days Allocated and Days Remaining

These fields indicate the number of days that were originally allocated for the license and how many days remain on the license. These fields are valid only for a time based license. For other licenses, the fields are not applicable ("n/a").

Languages Enabled

This pull down list shows all of the programming languages that are enabled for analysis by the license.

See the SAFE Corporation website for license costs, as they may change.

The Toolbar

The CodeSuite-LT toolbar is shown below.

BitMatch

This menu selection brings up the BitMatch form. See the section entitled Running BitMatch for more information.

CodeCross

This menu selection brings up the CodeCross form. See the section entitled Running CodeCross for more information.

CodeDiff

This menu selection brings up the CodeDiff form. See the section entitled Running CodeDiff for more information.

CodeMatch

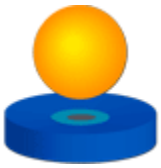
This menu selection brings up the CodeMatch form. See the section entitled Running CodeMatch for more information.

FileCount

This menu selection brings up the FileCount form. See the section entitled Running FileCount for more information.

FileIdentify

This menu selection brings up the FileIdentify form. See the section entitled Running FileIdentify for more information.



FileIsolate

This menu selection brings up the FileIsolate form. See the section entitled Running FileIsolate for more information.

Help

This menu selection brings up this users guide.

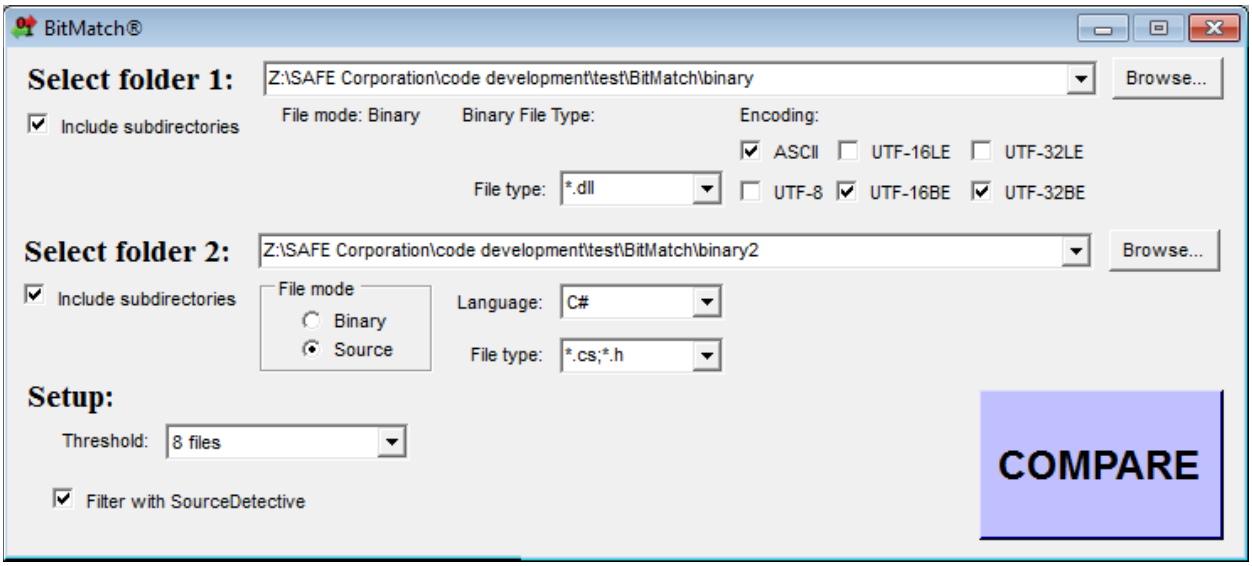
Authorize

This menu selection brings up the authorization form for entering licenses to enable the various tools. See the section entitled Licenses for more information.

BitMatch

Running BitMatch

BitMatch compares strings and identifier names from executable binary files to other executable binary files or source code files. Following that are step-by-step instructions for running BitMatch.



Step 1

Select the first folder for comparison by clicking on the browse button or entering the path in the text field. Check the box to include files in all subdirectories.

Step 2

Specify the types of binary files in the first folder to compare. You can type over the suggested file types with your own file types. Separate different file types with a semicolon. Use the * and ? wildcard characters if needed.

Step 3

Select the second folder for comparison by clicking on the browse button or entering the path in the text field. Check the box to include files in all subdirectories.

Step 4

Select the encoding of the files. If you know the encoding of the binary file, select it. If you do not know the encoding, you can use the following information to extract information from different possible encodings.

ASCII

ASCII encodes each character into a single byte. ASCII is used to extract text sequences based on printable character codes of decimal 32 to 126, plus horizontal tab. This mode was the default mode in prior versions of CodeSuite.

ASCII was used on DOS systems, early Linux systems, and many later systems. UTF-8 is common on Windows and Linux systems. Macintosh systems before OS X used an extension of ASCII called Mac OS Roman.

UTF-8

UTF-8 incorporates all of the ASCII characters, but provides for multi-byte character sequences to specify additional Unicode characters.

UTF-8 is popular for Internet browsing applications, which often require multi-language support. UTF-8 is also common on Windows, Linux, and Unix systems. Mac OS X uses UTF-8.

UTF-16LE and UTF-16BE

Both UTF-16 formats utilize sequences of two consecutive bytes to form a single 16-bit character. In UTF-16LE, ”LE” for ”little endian,” the low order byte occurs before the high order byte. In UTF-16BE, ”BE” for ”big endian,” the high byte precedes the low order byte.

UTF-16LE is common on Windows systems. UTF-16BE is common on Java platforms.

UTF-32LE and UTF-32BE

Both UTF-32 formats utilize four successive bytes to form a 32-bit character. In UTF-32LE, the bytes are sequenced from the low order byte to the high order byte. In UTF-16BE bytes are sequenced from the high order byte to the low order byte.

Step 5

Select whether the second folder contains binary files or source code files. If the second folder contains source code files, select the source code language.

Step 6

Specify the types of files in the second folder to compare. You can type over the suggested file types with your own file types. Separate different file types with a semicolon. Use the * and ? wildcard characters if needed.

Step 7

Select the reporting threshold from the pulldown menu. This determines how many files are reported. BitMatch reports only the most similar files. By setting the number of files to report to a large number you may get a very large report. By setting the number of files to report to a small number, the report will be smaller but it may not include all the similar files that you would like to see.

Step 8

Check the box to use SourceDetective® to filter out all matching elements that are found on the Internet, indicating that the matches are probably not due to copying but rather due to common usage.

Step 9

Click on the compare button. The number of licenses, if any, that are required for this run of BitMatch will be shown. You will have the ability to cancel the BitMatch run at this point without using up licenses.

You will be then asked for the name of the file and folder to contain the HTML reports.

BitMatch Algorithms

The Algorithms

BitMatch searches binary files for all uninterrupted sequences of text characters It then uses two CodeMatch algorithms to determine similarity between two source code files, first treating the text strings as program strings and then treating the text strings as identifiers. These algorithms are described below. When multiple files are compared, each match is given a weight and all weights are combined into a single matching score called the correlation score. The file pairs are then ranked by BitMatch score so that you can examine the most similar files.

Comment/string matching

BitMatch looks for identical comments and strings, ignoring whitespace. Comment lines and strings that contain only programming language keywords are still considered matches.

Identifier matching

BitMatch finds every instance in each file where identifiers match exactly. It eliminates programming language keywords and only reports matches for non-keyword identifiers such as variable names and function names.

BitMatch also finds every instance where an identifier in one file is part of a larger identifier in the other file. For example, the variable name "Index" in one file would partially match the variable names "NewIndex" and "Index1" in the other file. BitMatch eliminates programming language keywords and only reports matches for non-keyword identifiers such as variable names and function names.

Correlation Score

BitMatch produces a total correlation score based on the combination of above algorithms that the user chooses when running BitMatch . The minimum score is 0 while the maximum score is 100.

BitMatch Algorithms

The Algorithms

BitMatch searches binary files for all uninterrupted sequences of text characters It then uses two CodeMatch algorithms to determine similarity between two source code files, first treating the text strings as program strings and then treating the text strings as identifiers. These algorithms are described below. When multiple files are compared, each match is given a weight and all weights are combined into a single matching score called the correlation score. The file pairs are then ranked by BitMatch score so that you can examine the most similar files.

Comment/string matching

BitMatch looks for identical comments and strings, ignoring whitespace. Comment lines and strings that contain only programming language keywords are still considered matches.

Identifier matching

BitMatch finds every instance in each file where identifiers match exactly. It eliminates programming language keywords and only reports matches for non-keyword identifiers such as variable names and function names.

BitMatch also finds every instance where an identifier in one file is part of a larger identifier in the other file. For example, the variable name "Index" in one file would partially match the variable names "NewIndex" and "Index1" in the other file. BitMatch eliminates programming language keywords and only reports matches for non-keyword identifiers such as variable names and function names.

Correlation Score

BitMatch produces a total correlation score based on the combination of above algorithms that the user chooses when running BitMatch . The minimum score is 0 while the maximum score is 100.

BitMatch Basic Report

BitMatch Basic Report
Version: 1.0.1 | Date: 03/16/08 | Time: 15:35:00

SETTINGS | RESULTS | UNCOMPARED FILES | TOTALS

SETTINGS

Compare files in folder	C:\test\BitMatch\binary *Including subdirectories*
File types	*.exe
To files in folder	C:\test\BitMatch\C *Including subdirectories*
File types	*.c;*.h
Programming language	C
Reporting file threshold	8 files

RESULTS

C:\test\BitMatch\binary\CodeSuite.exe Score	Compared to file
27	C:\test\BitMatch\C\LineCount.c
23	C:\test\BitMatch\C\CodeSuite.h
21	C:\test\BitMatch\C\CodeSuite.c

C:\test\BitMatch\binary\test1\test.exe Score	Compared to file
22	C:\test\BitMatch\C\CodeSuite.c
21	C:\test\BitMatch\C\CodeSuite.h

C:\test\BitMatch\binary\test1\test0.exe Score	Compared to file
23	C:\test\BitMatch\C\CodeSuite.c
22	C:\test\BitMatch\C\LineCount.c
19	C:\test\BitMatch\C\CodeSuite.h

TOTALS

Total number of bytes in files in folder 1 = 799957
Total number of bytes in files in folder 2 = 292584
Total run time = 50 Seconds

TOP

BitMatch Detailed Report

BitMatch Detailed Report
Version: 1.0.1 | Date: 03/16/08 | Time: 15:35:00

SETTINGS

Compare file 1:	C:\test\BitMatch\binary\LineCount\Debug\LineCount.exe
To file 2:	C:\test\BitMatch\C\LineCount.c
Links to results:	Matching Comments and Strings Matching Identifiers Score

RESULTS

Matching Comments and Strings		
File1 Line#	**File2 Line#**	**Comment/String**
5890 9304 9381 9755 9765 10082 10093 10099 10199 10805 10815 14165	140	rB
14154	68	Total number of non-blank lines: %i
14157	65	Total number of Kbytes: %i

14158	64	Total number of files: %i

FILE	filepattern	folder	LineCount	name	size
stdio	string				

!CompareString	#File	(Press	.idata	arguments	ClHandle
DLineCount	Domain	Ession	files	FindFirstFile	IG_LINE
LCMapStringA	LoadLibrary	MultiByte	osfinfo.c	SandleCount	

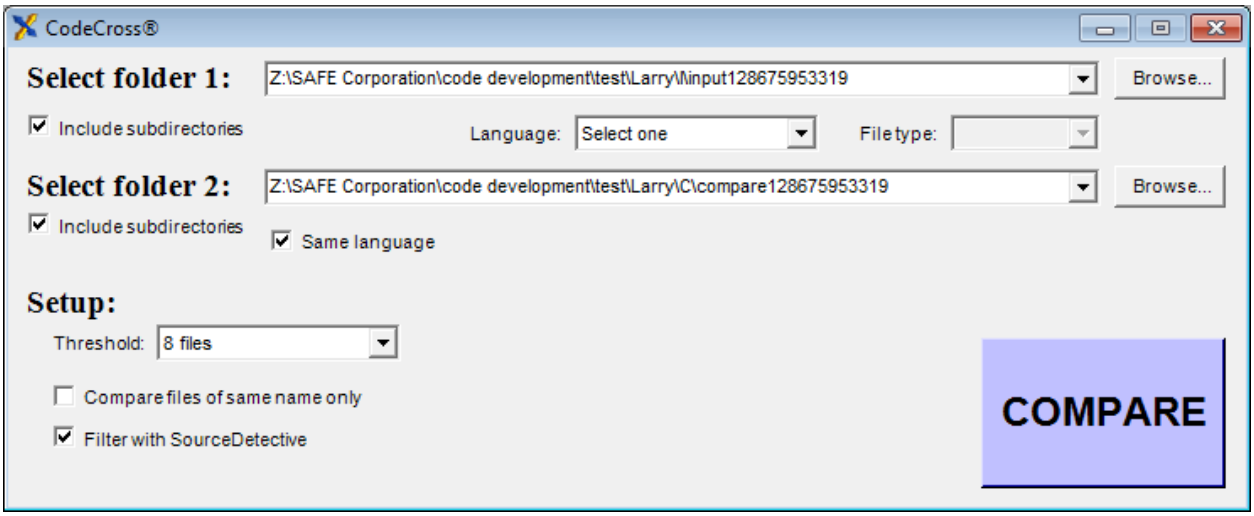
_A_SUBDIR	_finddata	_findfirst	_findnext	_LINE_LEN
finfo	handle	InString	IsBlank	KByteCount
SepString	stdlib			

SCORE 71

CodeCross

Running CodeCross

CodeCross cross-compares statements in one set of files to comments in the other set of files, and vice versa, in order to find code that has been commented out. CodeCross finds areas of source code that were used as guides to develop other source code; it finds signs of copying that CodeMatch can miss. Below is a screen shot of the CodeCross form. Following that are step-by-step instructions for running CodeCross.



Step 1

Select the first folder for comparison by clicking on the browse button or entering the path in the text field. Check the box to include files in all subdirectories.

Step 2

Select a source code language from the pulldown menu.

Step 3

Select the files types to compare from the pulldown menu. You can type over the suggested file types with your own file types. Separate multiple file types with a semicolon. Use the * and ? wildcard characters if needed.

Step 4

Select the second folder for comparison by clicking on the browse button or entering the path in the text field. Check the box to include files in all subdirectories.

Step 5

Choose whether both sets of files are written in the same programming language. If not, uncheck the box and you will be able to select the language and file types for the second set of files. CodeCross can compare files in different programming languages.

Step 6

Choose setup options to be used for comparing files.

Select the reporting threshold from the pulldown menu. This determines how many files are reported. CodeMatch reports only the most highly correlated files. By setting the number of files to report to a large number you may get a very large report. By setting the number of files to report to a small number, the report will be smaller but it may not include all the similar files that you would like to see.

Step 7

Choose whether to only compare files if they have the same name. This will speed up the comparison significantly because far fewer combinations of files are compared.

Step 8

Check the box to use SourceDetective® to filter out all matching elements that are found on the Internet, indicating that the matches are probably not due to copying but rather due to common usage.

Step 9

Click on the compare button. The number of licenses, if any, that are required for this run of CodeMatch will be shown. You will have the ability to cancel the CodeMatch run at this point without using up licenses.

You will be then asked for the name of the file and folder to contain the HTML reports.

CodeCross Algorithm

CodeCross compares statements in one file to comments and strings in another file and calculates the number of complete matches as a percentage of the total number of statements, comments, and strings.

CodeCross Score

CodeCross produces a score that is a combined percentage of statements that match comments and strings and a percentage of comments and strings that match statements. The minimum score is 0 while the maximum score is 100.

CodeCross Basic Report

CodeCross Basic Report
Version: 1.1.0 | Date: 12/29/08 | Time: 19:47:18

SETTINGS | RESULTS | UNCOMPARED FILES | TOTALS

SETTINGS

Compare files in folder	C:\test\CodeCross\files 1 *Including subdirectories*
File types	*.c;*.h
Programming language	C
To files in folder	C:\test\CodeCross\files 2 *Including subdirectories*
File types	*.c;*.h
Programming language	C
Reporting file threshold	8 files

RESULTS

C:\code development\test\CodeCross\files 1\bpf_dump_strings.c Score	Compared to file
71	C:\test\CodeCross\files 2\bpf_dump_identifers.c

71	C:\test\CodeCross\files 2\bpf_dump_mod.c
12	C:\test\CodeCross\files 2\aaa_commented.c
2	C:\test\CodeCross\files 2\W32NReg_commented.c

C:\code development\test\CodeCross\files 1\bpf_image.c Score	Compared to file
68	C:\test\CodeCross\files 2\bpf_dump_strings.c
2	C:\test\CodeCross\files 2\W32NReg_commented.c

C:\CodeCross\files 1\bpf_image_commented.c Score	Compared to file
38	C:\test\CodeCross\files 2\bpf_dump_strings.c
24	C:\test\CodeCross\files 2\W32NReg_commented.c

TOTALS

Total number of bytes in files in folder 1 = 33829
Total number of bytes in files in folder 2 = 25147
Total run time = 2 Seconds

TOP

CodeCross Detailed Report

CodeCross Detailed Report
Version: 1.1.0 | Date: 12/29/08 | Time: 19:47:18

SETTINGS

Compare file 1:	C:\test\CodeCross\files 1\aaa_case.c
To file 2:	C:\test\CodeCross\files 2\aaa_commented.c
Links to Results:	Matching Statements to Comments Matching Comments to Statements Score

RESULTS

Matching Statements to Comments		
File1 Line#	**File2 Line#**	**Statement**
1	1 4	P = Null;
2	2 5	Private String Auxonus = Null;

TOP

Matching Comments to Statements		
File1 Line#	**File2 Line#**	**Comment/String**
3	6	* The Regents of the University of California. All rights reserved.

5	8	* Redistribution and use in source and binary forms, with or without
6	9	* modification, are permitted provided that: (1) source code distributions
7	10	* retain the above copyright notice and this paragraph in its entirety, (2)
8	11	* distributions including binary code include the above copyright notice and
9	12	* this paragraph in its entirety in the documentation or other materials
10	13	* provided with the distribution, and (3) all advertising materials mentioning

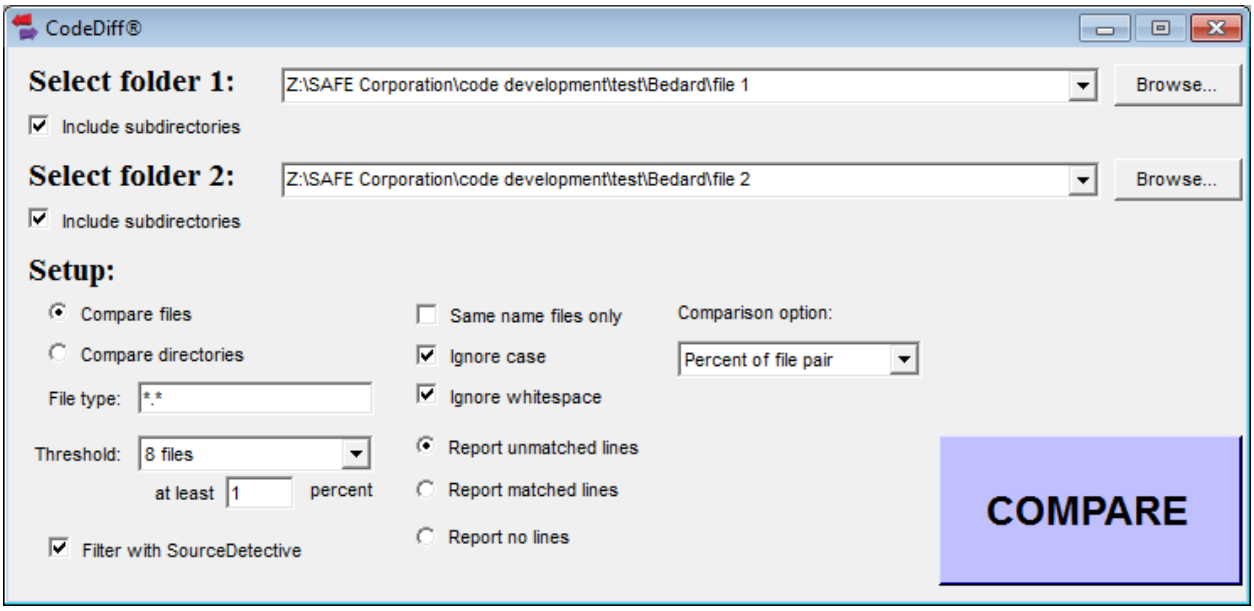
TOP

SCORE 100

CodeDiff

Running CodeDiff

CodeDiff compares files on a line by line basis to determine the percentage similarity. Below is a screen shot of the CodeDiff form. Following that are step-by-step instructions for running CodeDiff.



Step 1

Select the first folder for comparison by clicking on the browse button or entering the path in the text field. Check the box to include files in all subdirectories.

Step 2

Select the second folder for comparison by clicking on the browse button or entering the path in the text field. Check the box to include files in all subdirectories.

Step 3

Choose setup options to be used for comparing files.

You have a choice of two options.

- **Compare files.** CodeDiff will compare each pair of files in the directories specified and subdirectories if that option is selected. This option is used for comparing all combinations of files in all directories to find those that are similar.

- **Compare directories.** CodeDiff will compare only files with identical names in directories that have the same name and are in the same place in the directory tree. CodeDiff will also list all files in one directory that have no corresponding files in the other directory. This option is used for comparing entire directory trees to find similar files and missing files.

You have a choice of any combination of the following three options.

- **Same name files only.** CodeDiff will only compare files if they have the same exact name (not case sensitive). This option is not available when comparing directories, because only files of the same name are compared when comparing directories.
- **Ignore case.** CodeDiff will consider two lines matching if they are identical in all other respects even if the letter cases are different.
- **Ignore whitespace.** Before performing a comparison on any lines, CodeDiff will reduce all sequences of whitespace characters (space or tab) to a single space.

You have a choice of two options for reporting lines.

- **Report unmatched lines.** Report lines in either file that do not have a match in the other file. Note that if a line can be found three times in the first file and five times in the second file, two of the lines in the second file will be reported as unmatched.
- **Report matched lines.** Report lines that can be found in both files.
- **Report no lines.** Do not report any lines.

You have a choice of three comparison options.

- **Percentage of file pair.** The percentage generated by CodeDiff will be the percentage of lines in the two files that match with respect to the total number of lines in both files.
- **Percentage of first file.** The percentage generated by CodeDiff will be the percentage of lines in the first file that match a line in the second file with respect to the total number of lines in the first file.
- **Percentage of second file.** The percentage generated by CodeDiff will be the percentage of lines in the second file that match a line in the first file with respect to the total number of lines in the second file.

Step 4

Specify the types of files to compare. You can type over the suggested file types with your own file types. Separate different file types with a semicolon. Use the * and ? wildcard characters if needed.

Step 5

Select the reporting threshold from the pulldown menu. This determines how many files are reported. CodeDiff reports only the most similar files. By setting the number of files to report to a large number you may get a very large report. By setting the number of

files to report to a small number, the report will be smaller but it may not include all the similar files that you would like to see.

Specify the minimum percentage matching to report. CodeDiff reports the percentage of lines that are identical between files, ranging from 0 to 100. If you specify 0 as the minimum threshold, even files that do not have any matching lines will be reported. If you specify 100 as the minimum threshold, only files whose lines all match exactly will be reported.

Step 6

Check the box to use SourceDetective® to filter out all matching elements that are found on the Internet, indicating that the matches are probably not due to copying but rather due to common usage.

Step 7

Click on the compare button. The number of licenses, if any, that are required for this run of CodeDiff will be shown. You will have the ability to cancel the CodeDiff run at this point without using up licenses.

You will be then asked for the name of the file and folder to contain the HTML reports.

CodeDiff Basic Report

CodeDiff Basic Report
Version: 4.0.0 | Date: 03/14/09 | Time: 17:48:01

SETTINGS | RESULTS | UNCOMPARED FILES | TOTALS

SETTINGS

Compare files in folder	C:\test\C\files 1 *Including subdirectories*
File types	*.*
To files in folder	C:\test\C\files 2 *Including subdirectories*
File types	*.*
Algorithms selected	• Ignoring case • Ignoring whitespace • Percentage of file pairs • Report matched lines
Reporting file threshold	8 files
Reporting score threshold	1

RESULTS

C:\test\C\files 1\aaa.c Score	Compared to file
100	C:\test\C\files 2\aaa.c

80	C:\test\C\files 2\abc.c
3	C:\test\C\files 2\Copy of bpf_dump_strings.c

C:\test\C\files 1\aaa_case.c Score	Compared to file
100	C:\test\C\files 2\aaa.c
80	C:\test\C\files 2\abc.c
3	C:\test\C\files 2\bpf_dump_strings.c

C:\test\C\files 1\all_specifiers.c Score	Compared to file
100	C:\test\C\files 2\all_specifiers.c

TOTALS

Total number of bytes in files in folder 1 = 37651
Total number of bytes in files in folder 2 = 48944
Total run time = 3 Seconds

TOP

CodeDiff Detailed Report

CodeDiff Detailed Report
Version: 4.0.0 | Date: 03/14/09 | Time: 17:48:01

SETTINGS

Compare file 1:	C:\test\C\files 1\aaa_with_comments.c
To file 2:	C:\test\C\files 2\aaa_with_comments.c
Links to results:	Matched lines Score

RESULTS

Matching Lines		
File1 Line#	**File2 Line#**	**Line**
1	1	/* This is a comment*/ p = null;
2	2	private String auxonus = null; // This is a comment
3	3	p = /* This is a comment*/ null; /* This is a comment*/
4	4	/* Yes,
5	5	This is a comment*/ p = /* This is a comment*/ null; // This is a comment

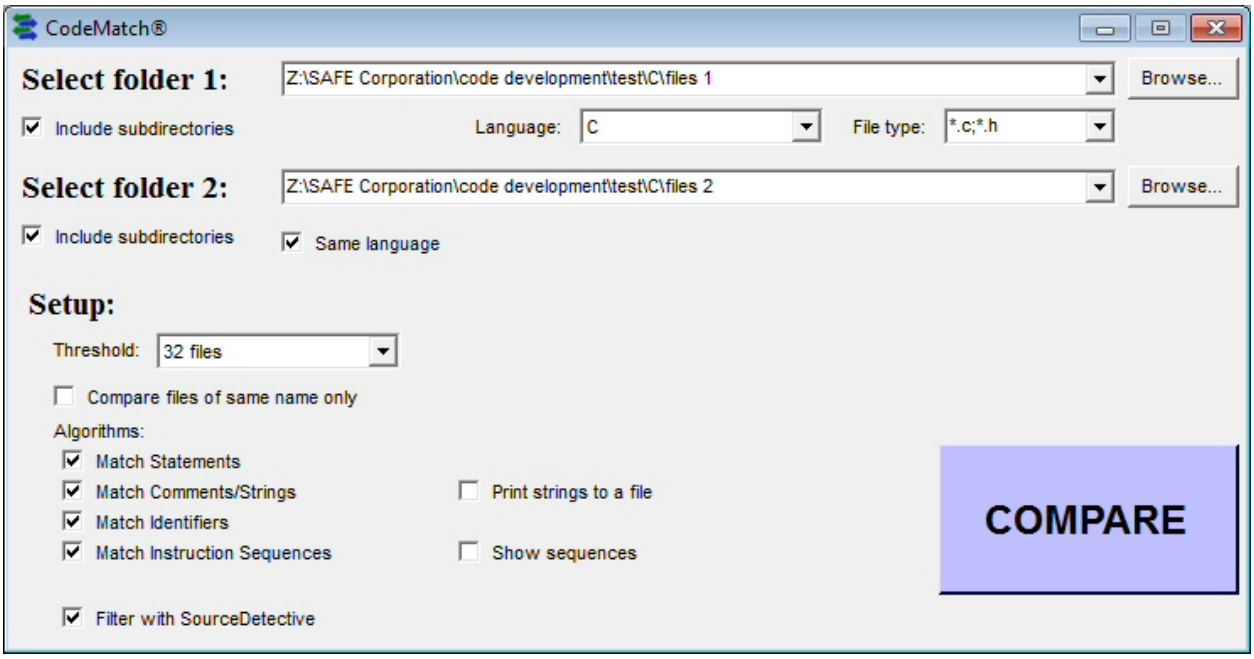
TOP

SCORE 100

CodeMatch

Running CodeMatch

CodeMatch compares files using a set of algorithms to determine their correlation. Below is a screen shot of the CodeMatch form. Following that are step-by-step instructions for running CodeMatch.



Step 1

Select the first folder for comparison by clicking on the browse button or entering the path in the text field. Check the box to include files in all subdirectories.

Step 2

Select a source code language from the pulldown menu.

Step 3

Select the files types to compare from the pulldown menu. You can type over the suggested file types with your own file types. Separate multiple file types with a semicolon. Use the * and ? wildcard characters if needed.

Step 4

Select the second folder for comparison by clicking on the browse button or entering the path in the text field. Check the box to include files in all subdirectories.

Step 5

Choose whether both sets of files are written in the same programming language. If not, uncheck the box and you will be able to select the language and file types for the second set of files. CodeMatch can compare files in different programming languages.

Step 6

Choose setup options to be used for comparing files.

Select the reporting threshold from the pulldown menu. This determines how many files are reported. CodeMatch reports only the most highly correlated files. By setting the number of files to report to a large number you may get a very large report. By setting the number of files to report to a small number, the report will be smaller but it may not include all the similar files that you would like to see.

Step 7

Choose whether to only compare files if they have the same name. This will speed up the comparison significantly because far fewer combinations of files are compared.

Step 8

Choose setup options and algorithms to be used for comparing files. You have a choice of any combination of four algorithms. For more information on each of these algorithms, see the section entitled CodeMatch Algorithms.

- **Statement Matching**
- **Comment Matching**
- **Identifier Matching**
- **Instruction Sequence Matching**

Note that when you select instruction sequence matching, you will have the option to report the instruction sequences. Reporting the sequences makes it easier to determine whether the sequences are interesting, but will produce a significantly larger report.

Also note that when you select comment matching, you will have the option to print strings into a file. This allows you to create a text file with a list of all strings that are found in both sets of code being compared.

Step 9

Check the box to use SourceDetective® to filter out all matching elements that are found on the Internet, indicating that the matches are probably not due to copying but rather due to common usage.

Step 10

Click on the compare button. The number of licenses, if any, that are required for this run of CodeMatch will be shown. You will have the ability to cancel the CodeMatch run at this point without using up licenses.

You will be then asked for the name of the file and folder to contain the HTML reports.

CodeMatch Algorithms

The Algorithms

CodeMatch uses several algorithms to determine similarity between two source code files. These algorithms are described below. When multiple files are compared, each match is given a weight and all weights are combined into a single matching score called the correlation score. The file pairs are then ranked by correlation score so that you can examine the most similar files.

Statement matching

CodeMatch looks for identical program statements (i.e., functional source code), ignoring whitespace and eliminating comments and strings. Statements that contain only programming language keywords are not considered matching. For statements to be considered matches, they must contain at least one identifier (non-keyword) such as a variable name or function name.

Comment/string matching

CodeMatch looks for identical comments and strings, ignoring whitespace. Comment lines and strings that contain only programming language keywords are still considered matches.

Instruction sequence matching

CodeMatch looks for sequences of instructions that match. CodeMatch notes the longest such sequence in each pair of files. A sequence matches if the initial programming language statement on each line is identical, regardless of what follows it. Even if variable names are altered in one file, CodeMatch will report similarities in the files. The following shows an example of two identical instruction sequences in C:

```
// File 1
if (x == 5)
{
    // Loop on j here
    for (j = 0; j < Index; j++)
        printf("x = %i", j);
}
else
    break;  // Here's the break

// File 2
if (xyz < 2)
    for (jjj = 0; jjj < i; jjj++)
    {
```

```
            printf("Hello world\n");
        }
    else
        break;
```

Identifier matching

CodeMatch finds every instance in each file where identifiers match exactly. It eliminates programming language keywords and only reports matches for non-keyword identifiers such as variable names and function names.

CodeMatch also finds every instance where an identifier in one file is part of a larger identifier in the other file. For example, the variable name "Index" in one file would partially match the variable names "NewIndex" and "Index1" in the other file. CodeMatch eliminates programming language keywords and only reports matches for non-keyword identifiers such as variable names and function names.

Correlation Score

CodeMatch produces a total correlation score based on the combination of above algorithms that the user chooses when running CodeMatch. The minimum score is 0 while the maximum score is 100.

CodeMatch Basic Report

CodeMatch Basic Report
Version: 5.3.1 | Date: 08/28/08 | Time: 11:33:11

SETTINGS | RESULTS | UNCOMPARED FILES | TOTALS

SETTINGS

Compare files in folder	C:\test\C\files 1 *Including subdirectories*
File types	*.c;*.h
Programming language	C
To files in folder	C:\test\C\files 2 *Including subdirectories*
File types	*.c;*.h
Programming language	C
Algorithms selected	• Statement Matching • Comment Matching • Identifier Matching • Instruction Sequence Matching
Reporting file threshold	8 files

RESULTS

C:\test\C\files 1\aaa.c	Compared to file

Score	
82	C:\test\C\files 2\aaa.c
82	C:\test\C\files 2\abc.c
71	C:\test\C\files 2\aaa_with_comments.c
15	C:\test\C\files 2\.svn\bpf_image.c
9	C:\test\C\files 2\all_specifiers.c
9	C:\test\C\files 2\bpf_dump_semicolons.c
9	C:\test\C\files 2\bpf_dump_strings.c
9	C:\test\C\files 2\semicolon_test.c

C:\test\C\files 1\aaa_case.c Score	Compared to file
82	C:\test\C\files 2\aaa.c
82	C:\test\C\files 2\abc.c
71	C:\test\C\files 2\aaa_with_comments.c
15	C:\test\C\files 2\.svn\W32NReg.c
13	C:\test\C\files 2\.svn\bpf_image.c

C:\test\C\files 1\aaa_whitespace.c Score	Compared to file
82	C:\test\C\files 2\aaa.c
82	C:\test\C\files 2\abc.c
15	C:\test\C\files 2\.svn\bpf_image.c
9	C:\test\C\files 2\all_specifiers.c

9	C:\test\C\files 2\bpf_dump_strings.c
9	C:\test\C\files 2\semicolon_test.c

C:\test\C\files 1\aaa_with_comments.c Score	Compared to file
87	C:\test\C\files 2\aaa_with_comments.c
71	C:\test\C\files 2\aaa.c
69	C:\test\C\files 2\abc.c
15	C:\test\C\files 2\.svn\bpf_image.c
8	C:\test\C\files 2\all_specifiers.c
8	C:\test\C\files 2\W32NReg.c

C:\test\C\files 1\all_ints.c Score	Compared to file
82	C:\test\C\files 2\all_ints.c
17	C:\test\C\files 2\all_specifiers.c
11	C:\test\C\files 2\.svn\W32NReg (no comments).c
11	C:\test\C\files 2\.svn\W32NReg.c

TOTALS

Total number of bytes in files in folder 1 = 37651
Total number of bytes in files in folder 2 = 37627
Total run time = 16 Seconds

CodeMatch Detailed Report

CodeMatch Detailed Report
Version: 5.3.1 | Date: 08/28/08 | Time: 11:33:11

SETTINGS

Compare file 1:	C:\test\C\files 1\bpf_image.c
To file 2:	C:\test\C\files 2\.svn\bpf_image.c
Links to results:	Matching Statements Matching Comments and Strings Matching Instruction Sequences Matching Identifiers Partially Matching Identifiers Score

RESULTS

Matching Statements

File1 Line#	File2 Line#	Statement
22	22	#include <windows.h>
23	23	#include <sys/types.h>
35	35	char *fmt, *op
36	36	static char image[256]
37	37	char operand[64]
39	39	v = p->k
40	40	switch (p->code) {

199 204 209 214	199	case BPF_ALU|BPF_OR|BPF_X:
254	254 259 264 269 270	case BPF_ALU|BPF_NEG:

TOP

Matching Comments and Strings

File1 Line#	File2 Line#	Comment/String
2	2	* Copyright (c) 1990, 1991, 1992, 1994, 1995, 1996
3	3	* The Regents of the University of California. All rights reserved.
5	5	* Redistribution and use in source and binary forms, with or without
6	6	* modification, are permitted provided that: (1) source code distributions
7	7	* retain the above copyright notice and this paragraph in its entirety, (2)
8	8	* distributions including binary code include the above copyright notice and
9	9	* this paragraph in its entirety in the documentation or other materials
10	10	* provided with the distribution, and (3) all advertising materials mentioning
11	11	* features or use of this software display the following acknowledgement:

TOP

Matching Instruction Sequences

File1 Line#	File2 Line#	Number of matching instructions
22	22	202
43	129	71
46	51	64
46	56	60
46	61	56
46	66	52
46	71	48
46	76	44
46	81	40
46	86	36
46	91	32

TOP

256	64	BPF_A	BPF_ABS	BPF_ADD	BPF_B
BPF_CLASS	BPF_DIV	BPF_H	bpf_image	BPF_IMM	BPF_JA
BPF_JEQ	BPF_JGE	BPF_JGT	BPF_JMP	BPF_JSET	BPF_LDX
BPF_LEN	BPF_LSH	BPF_MEM	BPF_MISC	BPF_MSH	BPF_OP
BPF_OR	BPF_RET	BPF_RSH	BPF_ST	BPF_STX	BPF_TXA
BPF_W	BPF_X	code	fmt	image	jt
op	operand	stdio	string	sys	

TOP

Partially Matching Identifiers
File1 Identifiers

0x00FF	BPF_ALU	bpf_filter	BPF_IMM	BPF_IND	BPF_LEN	BPF_MEMWORDS	BPF_RET
BPF_ST	BPF_SUB	EXTRACT_LONG	INT	netlong	types	UCHAR	W32N_htonl
winsock							

File2 Identifiers							
0x00004	0x00005	__stdcall	_TEXT	_W32N_ADA	_WAdapter0	_WAdapter1	_WAdapter2
dwDataLen	DWORD	dwType	ERR_IMPLIED	ERR_SUCCESS	H_LOCAL	hAdapter	hClassNet
KEY_READ	LONG	pAdapterInfo	PCHAR	PW_ADAPTER	QueryValue	TChar	VER_WIN32
W0Adapter	W0Window	W0Windows	W32N_Adapt	W32N_NET	WINCARDS	wsprintf	

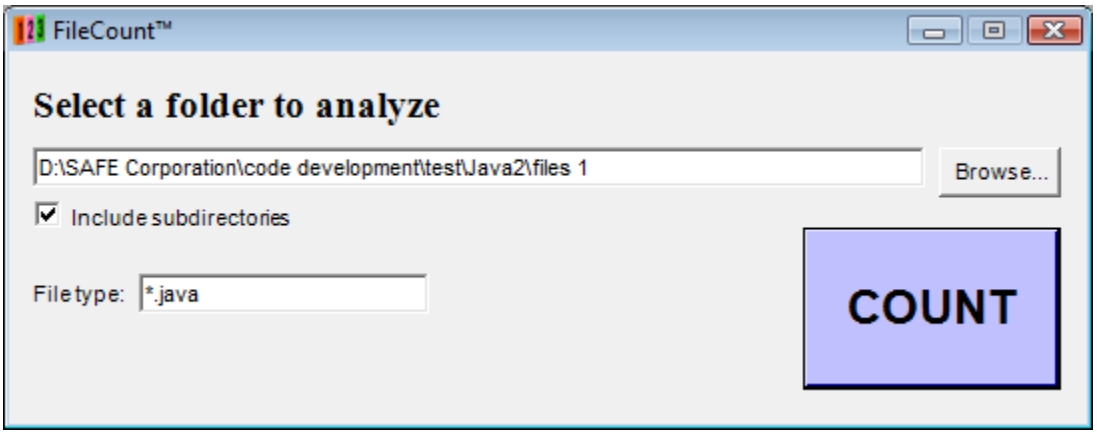
TOP

SCORE 100

FileCount

Running FileCount

FileCount is a utility that counts the number of files, non-blank lines, and bytes in a large set of files in a directory tree. FileCount is useful when using CodeDiff to generate statistics about a set of source code files.



Step 1

Select the folder where the files are that need to be counted by clicking on the browse button or entering the path in the text field. Check the box to include all subdirectories.

Step 2

Type in the file types. Separate different file types with a semicolon. Use the * and ? wildcard characters if needed.

Step 3

Press the count button. FileCount will then search the directory and all subdirectories, if specified, counting all of the files that meet the file type, and counting the total number of non-blank lines and bytes. When complete, a dialog box will appear with these counts.

FileIdentify

Running FileIdentify

FileIdentify allows a directory or directory tree and lists all of the file types found, based on the file name extensions. It also reports all known programming language files based on the file types. Below is a screen shot of the FileIdentify form and step-by-step instructions for running FileIdentify.

Step 1

Select the folder where the files are located that you want to analyze. Check the box to include all subdirectories if you want to analyze files in the subfolders also.

Step 2

Press the go button. You will be asked for the file name and location for a spreadsheet showing all file types and their associated programming languages, if known. FileIdentify will then search the directory and all subdirectories, if specified.

Below is an example of a spreadsheet created by FileIdentify.

	A	B	C
1	Analysis of Extensions		
2	Analysis date	12/16/2012	
3	Folder	Z:\SAFE Corporation\code development\test\FileIdentify\top	
4	Include subfolders	Yes	
5			
6	Files with no extension	0	
7	Files with an empty extension	0	
8	Folder paths too long	0	
9	File paths too long	0	
10			
11	File types	Number of files	Language (if known)
12	.as	37	ActionScript

13	.c	81	C
14	.cdb	6	
15	.csf	1	
16	.flr	1	
17	.gif	47	
18	.htm	181	
19	.jpg	8	
20	.js	413	JavaScript
21	.mako	1	
22	.php	8	PHP
23	.png	49	
24	.swf	517	
25	.txt	66	

The top line shows that the spreadsheet was an analysis of file extensions created by FileIdentify. The second line shows the date that the analysis was run. The third shows the folder name. The fourth line indicates whether or not subfolders were included in the analysis.

Line 6 gives the number of files that had no extension while line 7 gives the number of files that had an empty extension, meaning the file name ended in a dot. Line 8 gives the number of folders that exceeded the maximum number of characters and could thus not be examined while line 9 gives the number of file paths, meaning the folder name plus the file name, that exceeded the maximum number of characters and could thus not be examined.

Lines 12 through 25 show the files types that were found, in column A, the number of files for each file type, in column B, and the programming language, if known, in column C.

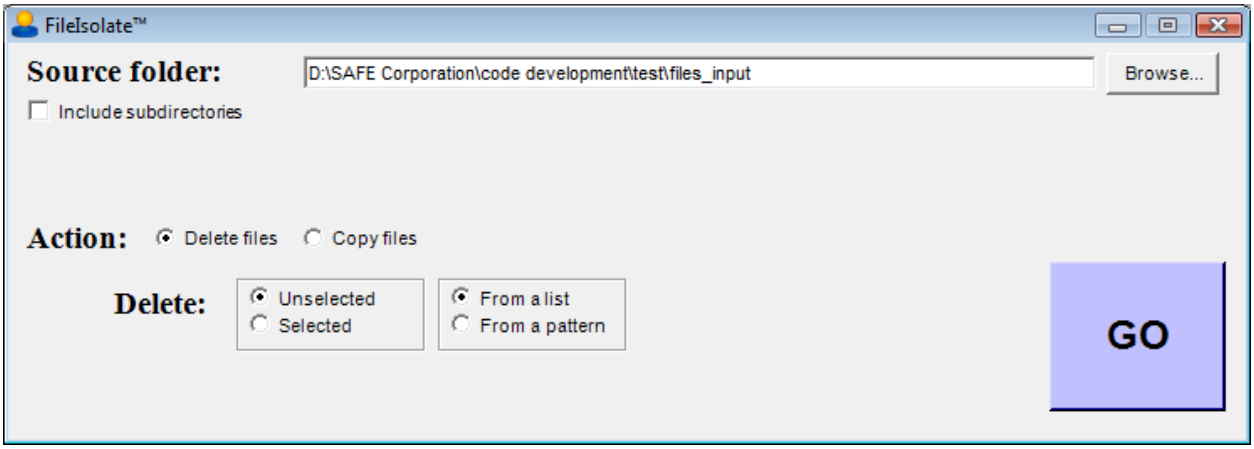
FileIsolate

Running FileIsolate

FileIsolate allows files and file types to be selectively copied or deleted from an entire directory tree.

Deleting files

Below is a screen shot of the FileIsolate form where the option is selected to delete files. Following that are step-by-step instructions for running FileIsolate to delete files.



Step 1

Choose the **Delete files** action.

Step 2

Select the folder where the files are that need to be deleted by clicking on the browse button or entering the path in the text field. Check the box to include all subdirectories.

Step 3

Choose options for deleting files.

- **Unselected files.** Choose this option to delete all files and file types that are not selected.
- **Selected files.** Choose this option to delete all files and file types that are selected.

Choose options for selecting files.

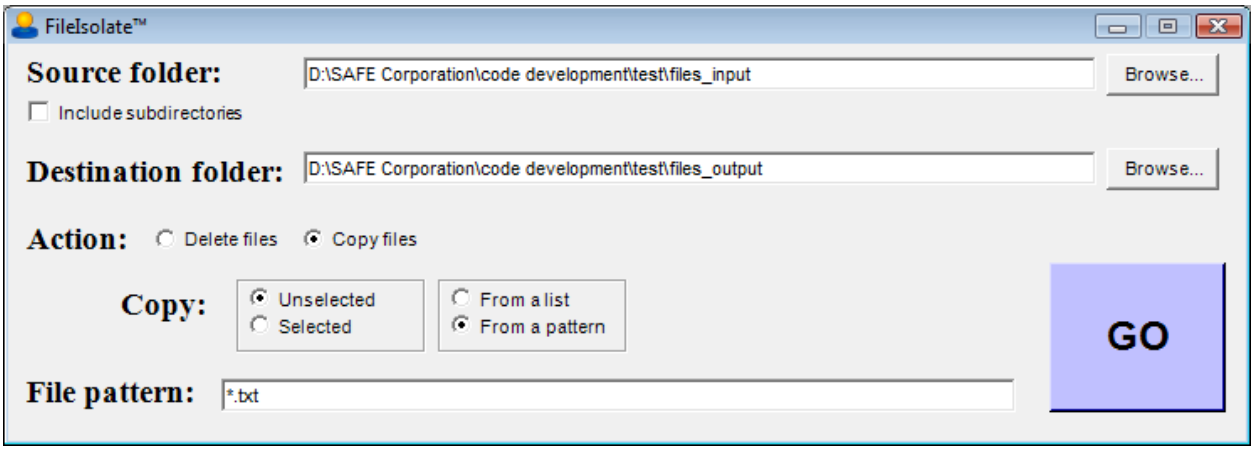
- **From a list of files.** Choose this option to select all files that are named in a text file. You will be prompted for the file containing the list of files. All files that have a name in this list will be selected.
- **From a file pattern.** Choose this option to select files whose names fit a pattern. A field will appear that allows you to type in file patterns. Separate different file types with a semicolon. Use the * and ? wildcard characters if needed.

Step 4

Press the go button. FileIsolate will then search the directory and all subdirectories, if specified. FileIsolate will delete all selected files or delete all files that were not selected, depending on the options specified.

Copying files

Below is a screen shot of the FileIsolate form where the option is selected to copy files. Following that are step-by-step instructions for running FileIsolate to copy files.



Step 1

Choose the **Copy files** action.

Step 2

Select the source folder where the files are that need to be copied by clicking on the browse button or entering the path in the text field. Check the box to include all subdirectories.

Step 3

Select the destination folder where the files are to be copied by clicking on the browse button or entering the path in the text field. If the destination folder does not exist, it will be created.

Step 4

Choose options for copying files.

- **Unselected files.** Choose this option to copy all files and file types that are not selected.
- **Selected files.** Choose this option to copy all files and file types that are selected.

Choose options for selecting files.

- **From a list of files.** Choose this option to select all files that are named in a text file. You will be prompted for the file containing the list of files. All files that have a name in this list will be selected.
- **From a file pattern.** Choose this option to select files whose names fit a pattern. A field will appear that allows you to type in file patterns. Separate different file types with a semicolon. Use the * and ? wildcard characters if needed.

Step 5

Press the go button. FileIsolate will then search the directory and all subdirectories, if specified. FileIsolate will copy all selected files or copy all files that were not selected, depending on the options specified.

Languages

Languages Supported

The following programming languages are currently supported:

ABAP	ASM-6502	ASM-65C02	ASM-65816	ASM-M68k
BASIC	C	C++	C#	COBOL
Delphi	DRI ASM	Flash ActionScript	Fortran	FoxPro
Go	Java	JavaScript	LISP	LotusScript
MASM	MATLAB	MPE/iX	Pascal	Perl
PHP	PL/M	PowerBuilder	PowerHouse	Prolog
Python	RealBasic	Ruby	Scala	SQL
TAL	TCL	Verilog	VHDL	Visual Basic

Check the SAFE Corporation website for new language modules, available at no charge, as they become available. If the language you need is not available, contact SAFE Corporation about creating it for a nominal fee.

Contacting SAFE Corporation

Software Analysis and Forensic Engineering Corporation
20863 Stevens Creek Blvd.
Suite 456
Cupertino, CA 95014
www.SAFE-Corp.com

Tel. (408) 517-1167
Fax (408) 693-3727
Email: Support@SAFE-Corp.com

Index