

**NAME**

cscout – C code analyzer and refactoring browser

**SYNOPSIS**

**cscout** [**-bCcErv3**] [**-d D**] [**-d H**] [**-l log file**] [**-p port**] [**-m specification**] [**-o | -s db**] [**-H proxy host**]  
 [**-P proxy host port**] [**-A username:password**] *file*

**DESCRIPTION**

*CScout* is a source code analyzer and refactoring browser for collections of C programs. It can process workspaces of multiple projects (we define a project as a collection of C source files that are linked together) mapping the complexity introduced by the C preprocessor back into the original C source code files. *CScout* takes advantage of modern hardware advances (fast processors and large memory capacities) to analyze C source code beyond the level of detail and accuracy provided by current compilers and linkers. The analysis *CScout* performs takes into account the identifier scopes introduced by the C preprocessor and the C language proper scopes and namespaces.

*CScout* as a source code analyzer can:

- annotate source code with hyperlinks to each identifier
- list files that would be affected by changing a specific identifier
- determine whether a given identifier belongs to the application or to an external library based on the accessibility and location of the header files that declare or define it
- locate unused identifiers taking into account inter-project dependencies
- perform queries for identifiers based on their namespace, scope, reachability, and regular expressions of their name and the filename(s) they are found in,
- perform queries for files, based on their metrics, or properties of the identifiers they contain
- monitor and report superfluously included header files
- provide accurate metrics on identifiers and files

More importantly, *CScout* helps you in refactoring code by identifying dead objects to remove, and can automatically perform accurate global *rename identifier* refactorings. *CScout* will automatically rename identifiers

- taking into account the namespace of each identifier: a renaming of a structure tag, member, or a statement label will not affect variables with the same name
- respecting the scope of the renamed identifier: a rename can affect multiple files, or variables within a single block, exactly matching the semantics the C compiler would enforce
- across multiple projects when the same identifier is defined in common shared include files
- occurring in macro bodies and *parts* of other identifiers, when these are created through the C preprocessor's token concatenation feature

This manual page describes the *CScout* invocation and command-line options. Details about its web interface, setup, and configuration can be found in the online hypertext documentation and at the project's home page <http://www.spinellis.gr/cscout>.

**OPTIONS**

- C** Create a ctags(1)-compatible tags file. Tens of editors and other tools can utilize tags to help you navigate through the code. In contrast to other tag-generation tools, the file that *CScout* generates includes information about entities generated through macros.
- c** Exit immediately after processing the specified files. Useful, when you simply want to check the source code for errors or when you want to create a *tags* file.

- d D** Display the `#define` directives being processed on the standard output.
- d H** Display the (mainly header) files being included on the standard output. Each line is prefixed by a number of dots indicating the depth of the included file stack.
- E** Preprocess the specified file and send the result to the standard output. Note that for this option to work correctly, you need to also process the workspace definition file with **-E**.
- p port** The web server will listen for requests on the TCP port number specified. By default the *CScout* server will listen at port 8081. The port number must be in the range 1024-32767.
- m specification**  
Specify the type of identifiers that *CScout* will monitor. The identifier attribute specification is given using the syntax: `Y|L|E|T[:attr1][:attr2]...` *The meaning of the first letter is:*
  - Y:** Match any of the specified attributes
  - L:** Match all of the specified attributes
  - E:** Exclude the specified attributes matched
  - T:** Exact match of the specified attributes

Allowable attribute names and their corresponding meanings are:

unused:

Unused identifier

writable:

Writable identifier

ro:

Read-only identifier

tag:

Tag for a struct/union/enum

member:

Member of a struct/union

label:

Label

obj:

Ordinary identifier (note that enumeration constants and typedefs belong to the ordinary identifier namespace)

macro:

Preprocessor macro

umacro:

Undefined preprocessor macro

macroarg:

Preprocessor macro argument

fscope:

Identifier with file scope

pscope:

Identifier with project scope

typedef:

Typedef

enumconst:

Enumeration constant

The **-m** flag can provide enormous savings on the memory *CScout* uses (specify e.g. **-m Y:pscope** to only track project-global identifiers), but the processing *CScout* performs under this flag is *unsound*. The flag should therefore be used only if you are running short of memory. There are cases where the use of preprocessor macros can change the attributes of a given identifier shared between different files. Since the **-m** optimization is performed after each single file is processed, the locations where an identifier is found may be misrepresented.

- r** Report on the standard error output warnings about unused and wrongly scoped identifiers and unused included files. The error message format is compatible with *gcc* and can therefore be automatically processed by editors that recognize this format.
- v** Display the *CScout* version and copyright information and exit.
- 3** Implement support for trigraph characters.

## MORE OPTIONS

The following options are only available on the version of *CScout* available through a support license.

- b** Operate in multiuser browse-only mode. In this mode the web server can concurrently process multiple requests. All web operations that can affect the server's functioning (such as setting the various options, renaming identifiers, refactoring function arguments, selecting a project, editing a file, or terminating the server) are prohibited. Call graphs are truncated to 1000 elements (nodes or edges).
- H** *proxy host*  
Specify a proxy HTTP host for connecting to the program's auditing and licensing server.
- P** *proxy host port*  
Specify the proxy HTTP host port for connecting to the program's auditing and licensing server. The default port is 80.
- s** *database dialect*  
Dump the workspace contents as an SQL script. Specify *help* as the database dialect to obtain a list of supported database back-ends.
- A** *username:password*  
Specify a proxy host authorization username and password for connecting to the program's auditing and licensing server.
- l** *log file*  
Specify the location of a file where web requests will be logged.
- o** Create obfuscated versions of all the writable files of the workspace.

## EXAMPLE

Assume you want to analyze three programs in `/usr/src/bin`. You first create the following project definition file, `bin.prj`.

```
# Some small tools from the src/bin directory
workspace bin {
    ro_prefix "/usr/include"
    cd "/usr/src/bin"
    project cp {
        cd "cp"
        file cp.c utils.c
    }
    project echo {
        cd "echo"
        file echo.c
    }
    project date {
        cd "date"
        file date.c
    }
}
```

Then you compile the workspace file `bin.prj` by running the *CScout* workspace compiler *cswc* on it, and finally you run *cscout* on the compiled workspace file. At that point you are ready to analyze your code and rename its identifiers through your web browser.

```
$ cswc bin.prj >bin.cs
$ cscout bin.cs
Processing workspace bin
Entering directory /usr/src/bin
Processing project cp
Entering directory cp
Processing file cp.c
Done processing file cp.c
Processing file utils.c
Done processing file utils.c
Exiting directory cp
Done processing project cp
Processing project echo
Entering directory echo
Processing file echo.c
Done processing file echo.c
Exiting directory echo
Done processing project echo
Processing project date
Entering directory date
Processing file date.c
Done processing file date.c
Exiting directory date
Done processing project date
Exiting directory /usr/src/bin
Done processing workspace bin
Post-processing /usr/home/dds/src/cscout/bin.c
[...]
Post-processing /vol/src/bin/cp/cp.c
Post-processing /vol/src/bin/cp/extern.h
Post-processing /vol/src/bin/cp/utils.c
Post-processing /vol/src/bin/date/date.c
Post-processing /vol/src/bin/date/extern.h
Post-processing /vol/src/bin/date/vary.h
Post-processing /vol/src/bin/echo/echo.c
Processing identifiers
100%
We are now ready to serve you at http://localhost:8081
```

**SEE ALSO**

cswc(1)

**AUTHOR**

(c) Copyright 2003-2010 Diomidis Spinellis.