C Code Refactoring

Working with CScout, the C Refactoring Browser

Diomidis Spinellis

www.spinellis.gr/dds

Problems with C and C++

- Macros complicate the meaning of:
  - Identifiers
    - We can create new ones!
  - Scope
    - Extended borders
    - Intermingled namespaces
    - Merged files
- This makes tool-based refactoring tricky

Namespace Intermingling

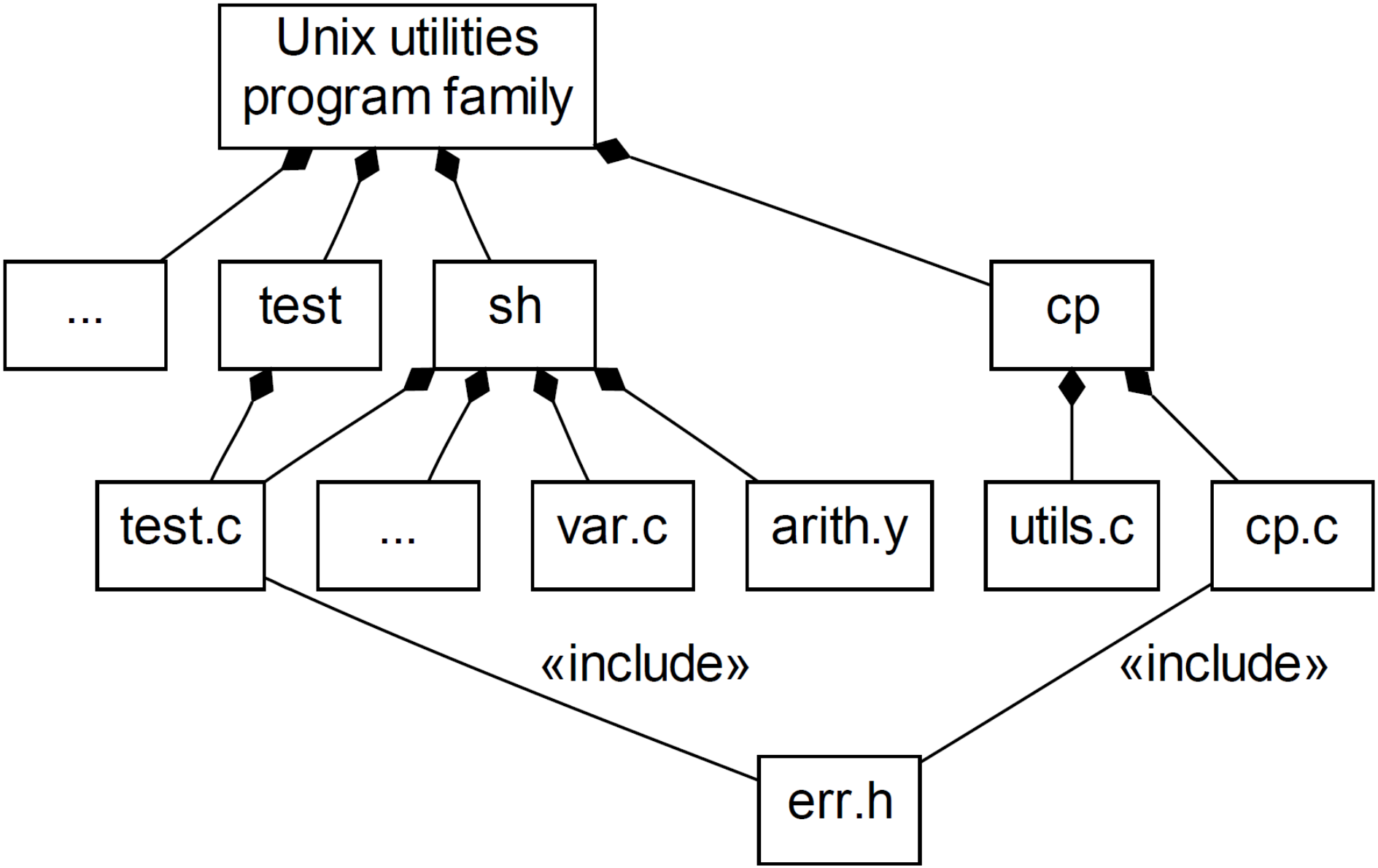
```
struct a { int field; };
struct b { int field; };
#define getfield(var) (var.field)

int foo(void *p) {
  struct a sa;
  struct b sb;

  return (getfield(sa) + getfield(sb));
}
```

Name Creation

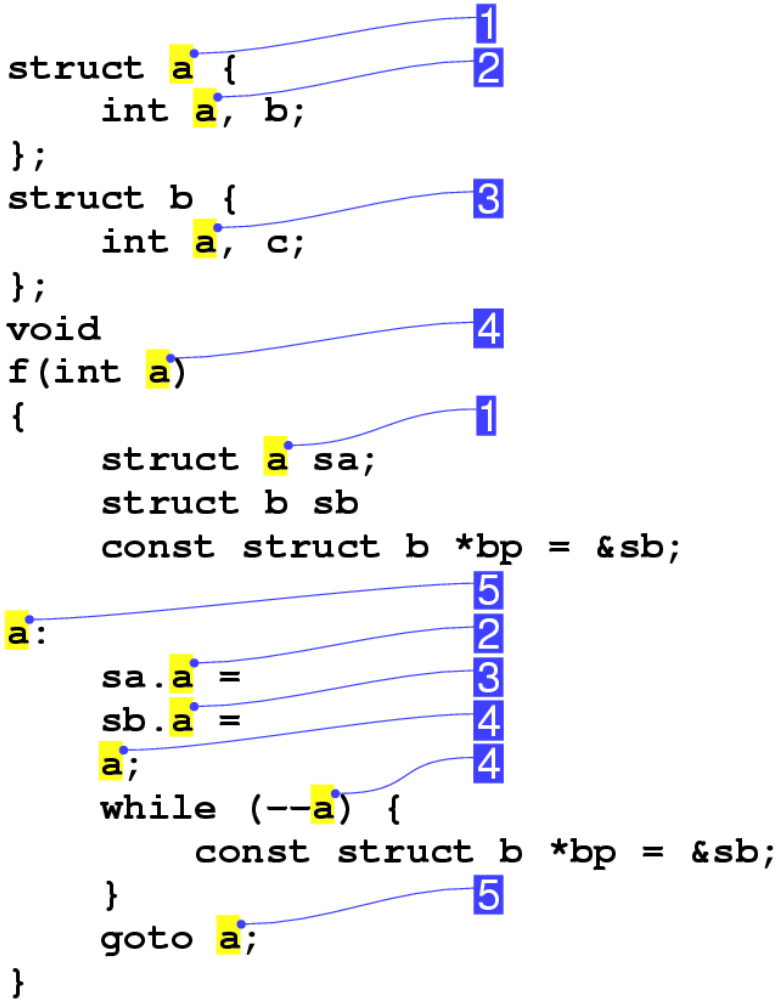
```
#define typefun(name, type, op) \
type type ## _ ## name(type a, type b) {\
  return a op b; }
typefun(add, int, +)
typefun(sub, int, -)
typefun(add, double, +)
typefun(sub, double, -)
main() {
  printf("%d\n", int_add(5, 4));
  printf("%g\n", double_sub(3.14, 2.0));
}
```


Unix utilities
program family
...
test
sh
cp
test.c
...
var.c
arith.y
utils.c
cp.c
«include»
«include»
err.h

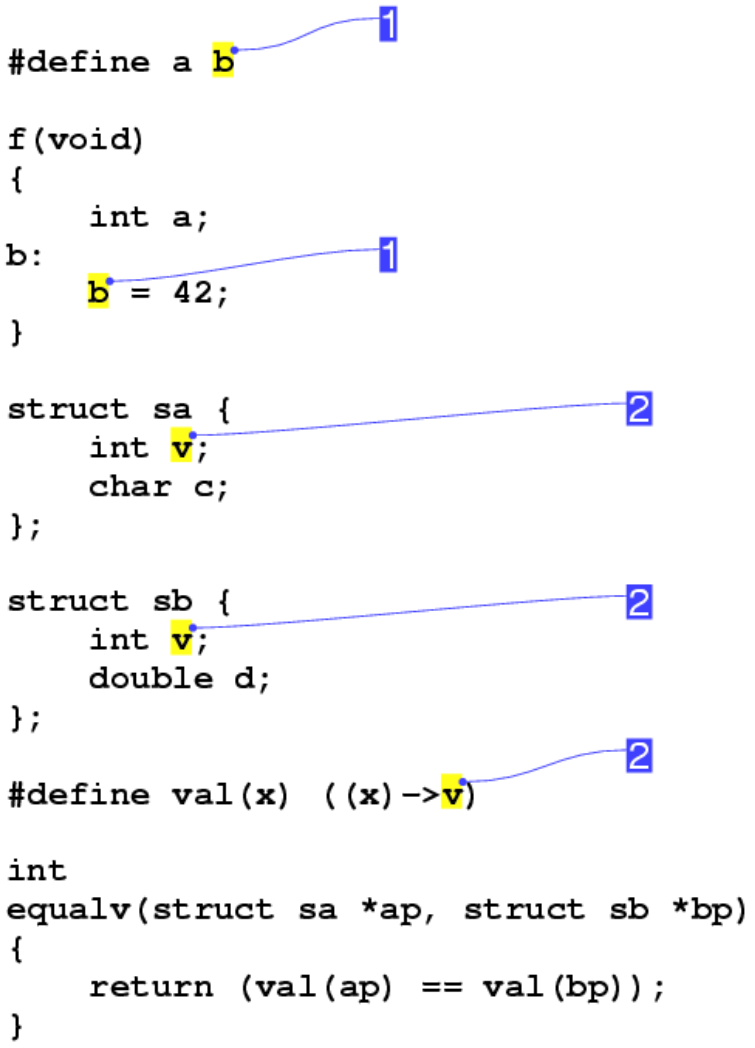
Refactoring Strategy

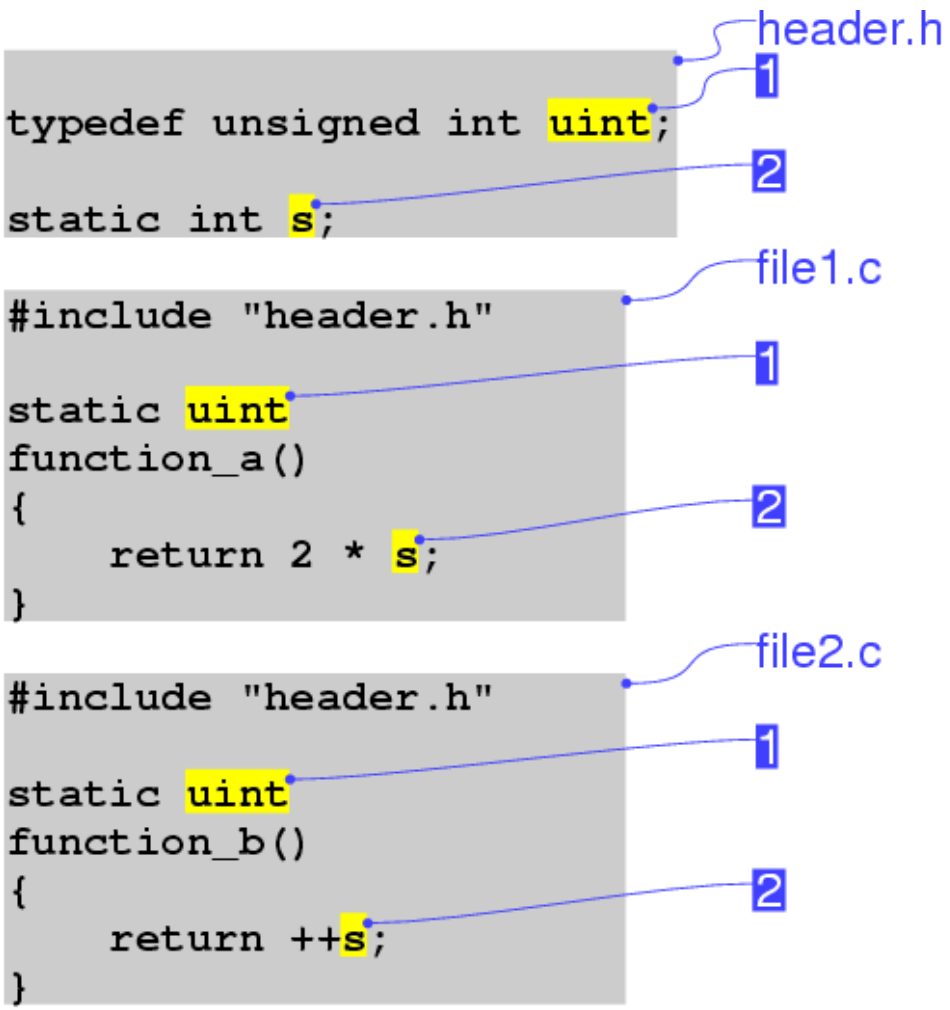
- Analyze with preprocessor
- Determine equivalence classes
  - Semantic equivalence
  - Lexical equivalence
  - Partial lexical equivalence

Lexical Equivalence

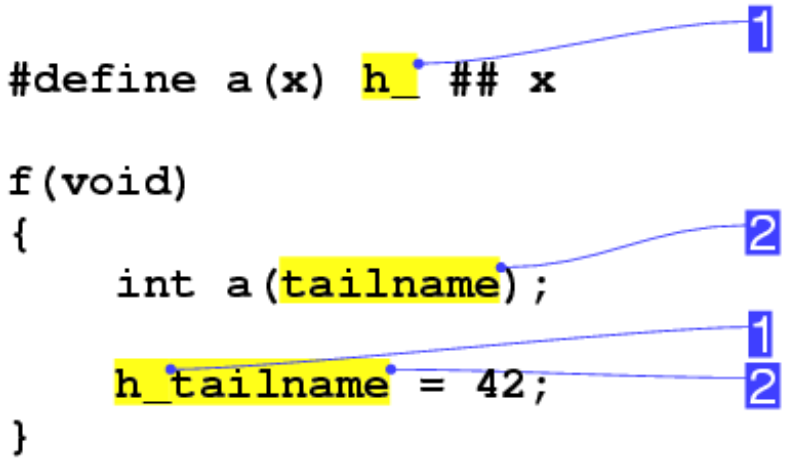


Lexical Equivalence





Partial Lexical Equivalence



Problems and Solutions

- Multiple configurations (#ifdef)
  - Multiple processing
- Library code
  - Check access permissions
- Source code reuse
  - Process all systems together

The Cscout Refactoring Browser

- Browsing and analysis of files and names
- Web based – can be used by teams
- Code becomes hypertext
- Call graph
- Generalized queries
- Variable substitution
- Metrics
- Code obfuscation
- Database interface

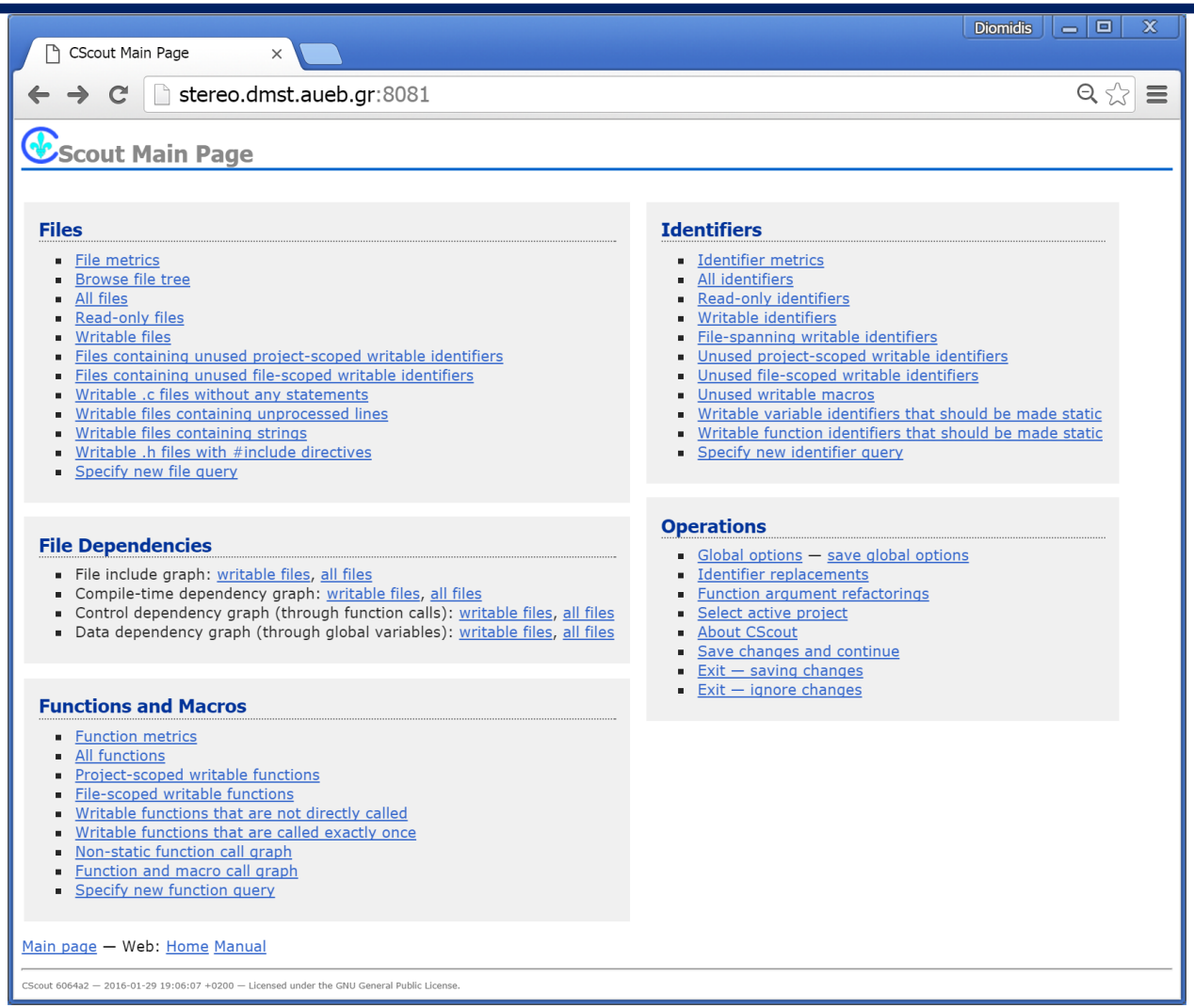
Walkthrough (canned)

```
$ git clone https://github.com/dspinellis/cscout.git
$ cd cscout
$ make
$ make test # optional
$ sudo make install
$ cd example
$ cscout awk.cs
CScout is now ready to serve you at http://localhost:8081
```

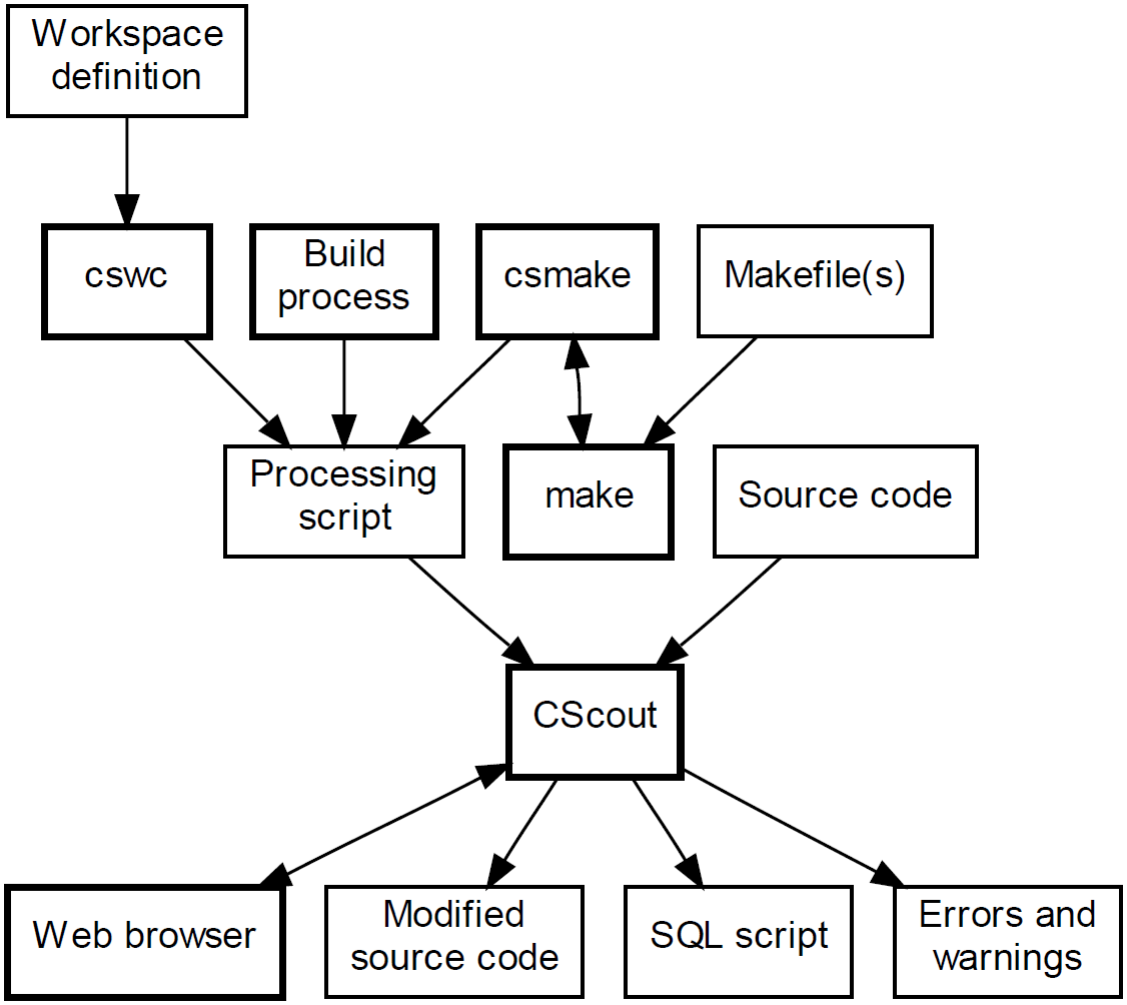
Walkthrough (on your code)

```
$ make clean
$ csmake
$ cscout make.cs
CScout is now ready to serve you at http://localhost:8081
```

Workspace

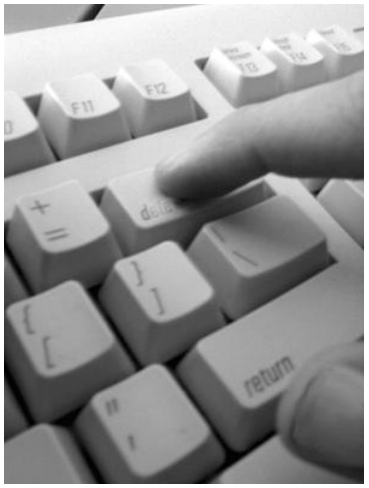

Diomidis
CScout Main Page
stereo.dmst.aueb.gr:8081
Scout Main Page
Files
File metrics
Browse file tree
All files
Read-only files
Writable files
Files containing unused project-scoped writable identifiers
Files containing unused file-scoped writable identifiers
Writable .c files without any statements
Writable files containing unprocessed lines
Writable files containing strings
Writable .h files with #include directives
Specify new file query
File Dependencies
File include graph: writable files, all files
Compile-time dependency graph: writable files, all files
Control dependency graph (through function calls): writable files, all files
Data dependency graph (through global variables): writable files, all files
Functions and Macros
Function metrics
All functions
Project-scoped writable functions
File-scoped writable functions
Writable functions that are not directly called
Writable functions that are called exactly once
Non-static function call graph
Function and macro call graph
Specify new function query
Identifiers
Identifier metrics
All identifiers
Read-only identifiers
Writable identifiers
File-spanning writable identifiers
Unused project-scoped writable identifiers
Unused file-scoped writable identifiers
Unused writable macros
Writable variable identifiers that should be made static
Writable function identifiers that should be made static
Specify new identifier query
Operations
Global options — save global options
Identifier replacements
Function argument refactorings
Select active project
About CScout
Save changes and continue
Exit — saving changes
Exit — ignore changes
Main page — Web: Home Manual
CScout 6064a2 — 2016-01-29 19:06:07 +0200 — Licensed under the GNU General Public License.

How it all fits together



Refactoring queries

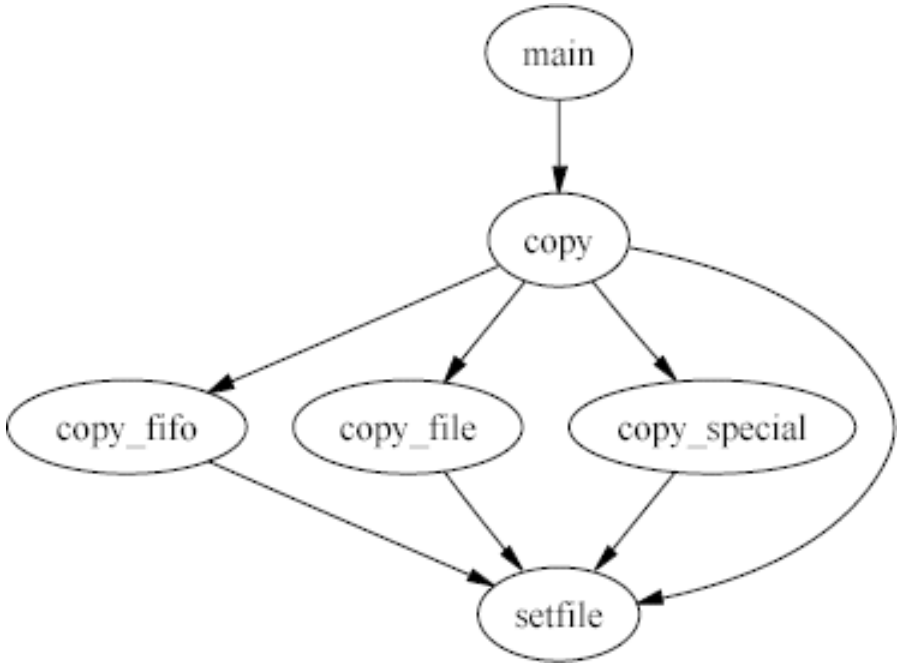
- Non-used elements
  - Functions
  - Variables
  - Structure fields
  - Macros
  - Included files
- Can probably be deleted



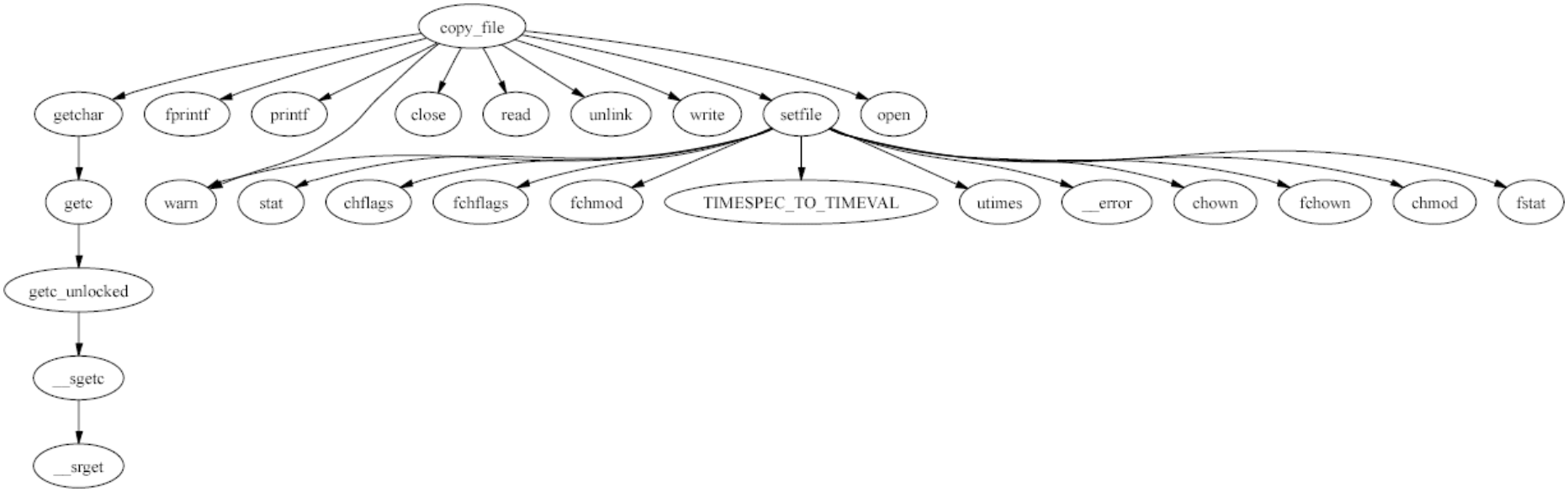
Refactoring Queries

- Functions not called
  - Dead code;
- Functions called once
  - Inlining

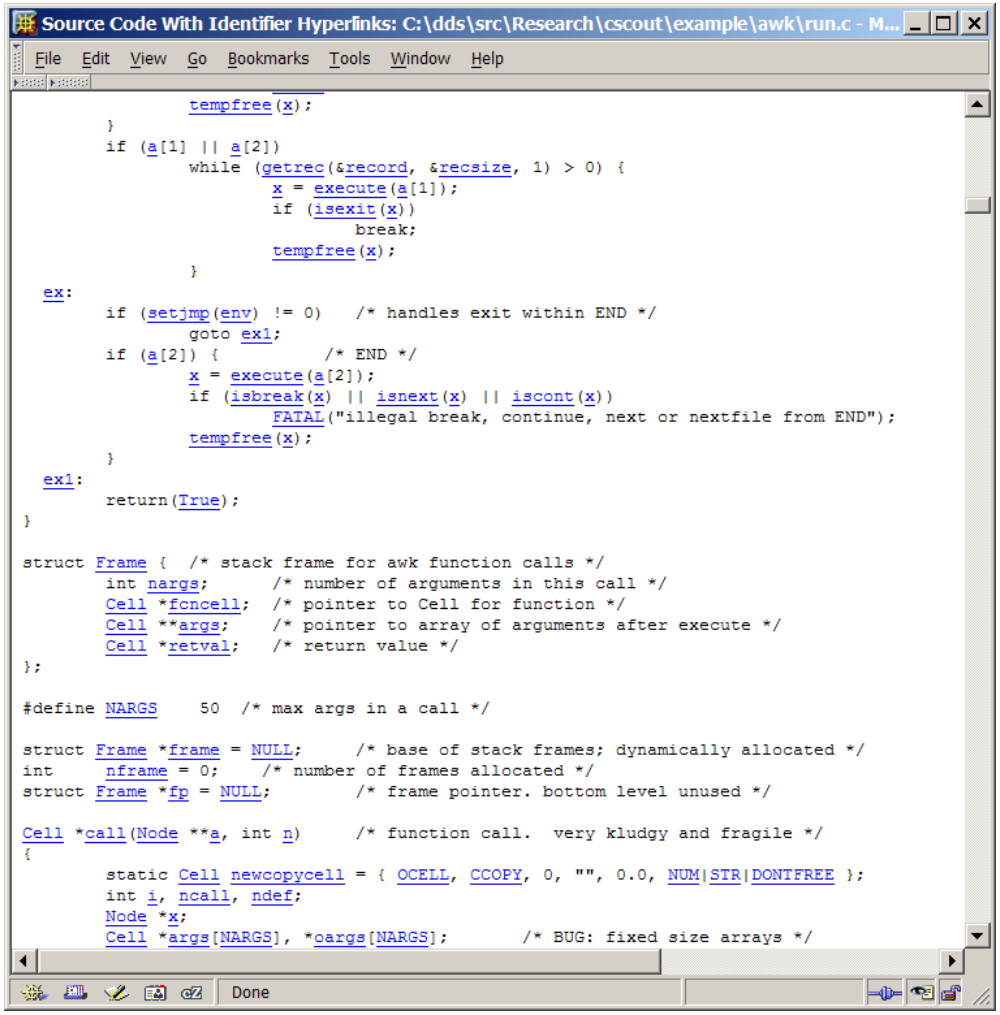
Call Graph



Caller Graph



Code Browsing




```
                tempfree(x);
        }
        if (a[1] || a[2])
                while (getrec(&record, &recsize, 1) > 0) {
                        x = execute(a[1]);
                        if (isexit(x))
                                break;
                        tempfree(x);
                }
  ex:
        if (setjmp(env) != 0)   /* handles exit within END */
                goto ex1;
        if (a[2]) {             /* END */
                x = execute(a[2]);
                if (isbreak(x) || isnext(x) || iscont(x))
                        FATAL("illegal break, continue, next or nextfile from END");
                tempfree(x);
        }
  ex1:
        return(True);
}

struct Frame {  /* stack frame for awk function calls */
        int nargs;      /* number of arguments in this call */
        Cell *fncell;   /* pointer to Cell for function */
        Cell **args;    /* pointer to array of arguments after execute */
        Cell *retval;   /* return value */
};

#define NARGS    50  /* max args in a call */

struct Frame *frame = NULL;     /* base of stack frames; dynamically allocated */
int     nframe = 0;     /* number of frames allocated */
struct Frame *fp = NULL;        /* frame pointer. bottom level unused */

Cell *call(Node **a, int n)     /* function call.  very kludgy and fragile */
{
        static Cell newcopycell = { OCELL, CCOPY, 0, "", 0.0, NUM|STR|DONTFREE };
        int i, ncall, ndef;
        Node *x;
        Cell *args[NARGS], *oargs[NARGS];       /* BUG: fixed size arrays */
```

Name Details

Identifier: execute

- Read-only: No
- Tag for struct/union/enum: No
- Member of struct/union: No
- Label: No
- Ordinary identifier: Yes
- Macro: No
- Undefined macro: No
- Macro argument: No
- File scope: No
- Project scope: Yes
- Typedef: No
- Enumeration constant: No
- Yacc identifier: No
- Function: Yes
- Crosses file boundary: Yes
- Unused: No
- Matches 83 occurence(s)
- Appears in project(s):
  - maketab
  - a.out
  - /home/dds/src/cscout/example/awk/a.out
- Dependent files
- Associated functions
- The identifier occurs (wholy or in part) in function name(s):
  1. [execute] — function page
- Substitute with: execute Save

Main page — Web: Home Manual

Generalized Queries

- Names
- Files
- Functions

Identifier Query

- [x] Writable
- [ ] Read-only
- [ ] Tag for struct/union/enum
- [ ] Member of struct/union
- [ ] Label
- [ ] Ordinary identifier
- [ ] Macro
- [ ] Undefined macro
- [ ] Macro argument
- [ ] File scope
- [ ] Project scope
- [ ] Typedef
- [ ] Enumeration constant
- [ ] Yacc identifier
- [ ] Function
- [ ] Crosses file boundary
- [ ] Unused

(•) Match any marked ( ) Match all marked ( ) Exclude marked ( ) Exact match

---

Identifier names should (☐ not) match RE execute

Select identifiers from filenames (☐ not) matching RE

Query title [ ] Show identifiers | Show files | Show functions

Main page — Web: Home Manual

File Query

☐ Writable
☐ Read-only

Sort-by	Metric	Compare	Value
○	Number of characters	ignore ▼	
○	Number of comment characters	ignore ▼	
○	Number of space characters	ignore ▼	
○	Number of line comments	ignore ▼	
○	Number of block comments	ignore ▼	
○	Number of lines	ignore ▼	
○	Maximum number of characters in a line	ignore ▼	
○	Number of character strings	ignore ▼	
○	Number of unprocessed lines	ignore ▼	
○	Number of C preprocessor directives	ignore ▼	
○	Number of processed C preprocessor conditionals (ifdef, if, elif)	ignore ▼	
○	Number of defined C preprocessor function-like macros	ignore ▼	
○	Number of defined C preprocessor object-like macros	ignore ▼	
○	Number of preprocessed tokens	ignore ▼	
○	Number of compiled tokens	ignore ▼	
○	Number of copies of the file	ignore ▼	
○	Number of statements	ignore ▼	
○	Number of defined project-scope functions	ignore ▼	
○	Number of defined file-scope (static) functions	ignore ▼	
○	Number of defined project-scope variables	ignore ▼	
○	Number of defined file-scope (static) variables	ignore ▼	
○	Number of complete aggregate (struct/union) declarations	ignore ▼	
○	Number of declared aggregate (struct/union) members	ignore ▼	
○	Number of complete enumeration declarations	ignore ▼	
○	Number of declared enumeration elements	ignore ▼	
○	Number of directly included files	ignore ▼	
◉	Entity name		

☐ Reverse sort order

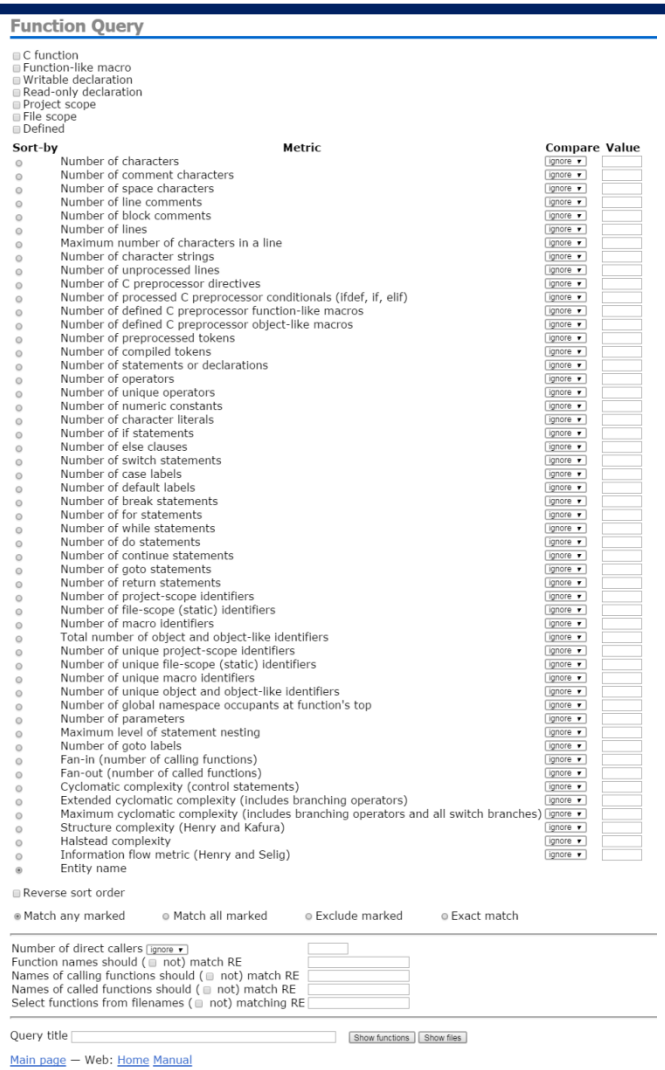
◉ Match any of the above ○ Match all of the above

File names should (☐ not) match RE

Query title Show files

Main page — Web: Home Manual

Function Query



Function Query

C function
Function-like macro
Writable declaration
Read-only declaration
Project scope
File scope
Defined

Sort-by	Metric	Compare	Value
○	Number of characters	ignore	
○	Number of comment characters	ignore	
○	Number of space characters	ignore	
○	Number of line comments	ignore	
○	Number of block comments	ignore	
○	Number of lines	ignore	
○	Maximum number of characters in a line	ignore	
○	Number of character strings	ignore	
○	Number of unprocessed lines	ignore	
○	Number of C preprocessor directives	ignore	
○	Number of processed C preprocessor conditionals (ifdef, if, elif)	ignore	
○	Number of defined C preprocessor function-like macros	ignore	
○	Number of defined C preprocessor object-like macros	ignore	
○	Number of preprocessed tokens	ignore	
○	Number of compiled tokens	ignore	
○	Number of statements or declarations	ignore	
○	Number of operators	ignore	
○	Number of unique operators	ignore	
○	Number of numeric constants	ignore	
○	Number of character literals	ignore	
○	Number of if statements	ignore	
○	Number of else clauses	ignore	
○	Number of switch statements	ignore	
○	Number of case labels	ignore	
○	Number of default labels	ignore	
○	Number of break statements	ignore	
○	Number of for statements	ignore	
○	Number of while statements	ignore	
○	Number of do statements	ignore	
○	Number of continue statements	ignore	
○	Number of goto statements	ignore	
○	Number of return statements	ignore	
○	Number of project-scope identifiers	ignore	
○	Number of file-scope (static) identifiers	ignore	
○	Number of macro identifiers	ignore	
○	Total number of object and object-like identifiers	ignore	
○	Number of unique project-scope identifiers	ignore	
○	Number of unique file-scope (static) identifiers	ignore	
○	Number of unique macro identifiers	ignore	
○	Number of unique object and object-like identifiers	ignore	
○	Number of global namespace occupants at function's top	ignore	
○	Number of parameters	ignore	
○	Maximum level of statement nesting	ignore	
○	Number of goto labels	ignore	
○	Fan-in (number of calling functions)	ignore	
○	Fan-out (number of called functions)	ignore	
○	Cyclomatic complexity (control statements)	ignore	
○	Extended cyclomatic complexity (includes branching operators)	ignore	
○	Maximum cyclomatic complexity (includes branching operators and all switch branches)	ignore	
○	Structure complexity (Henry and Kafura)	ignore	
○	Halstead complexity	ignore	
○	Information flow metric (Henry and Selig)	ignore	
◉	Entity name		

Reverse sort order

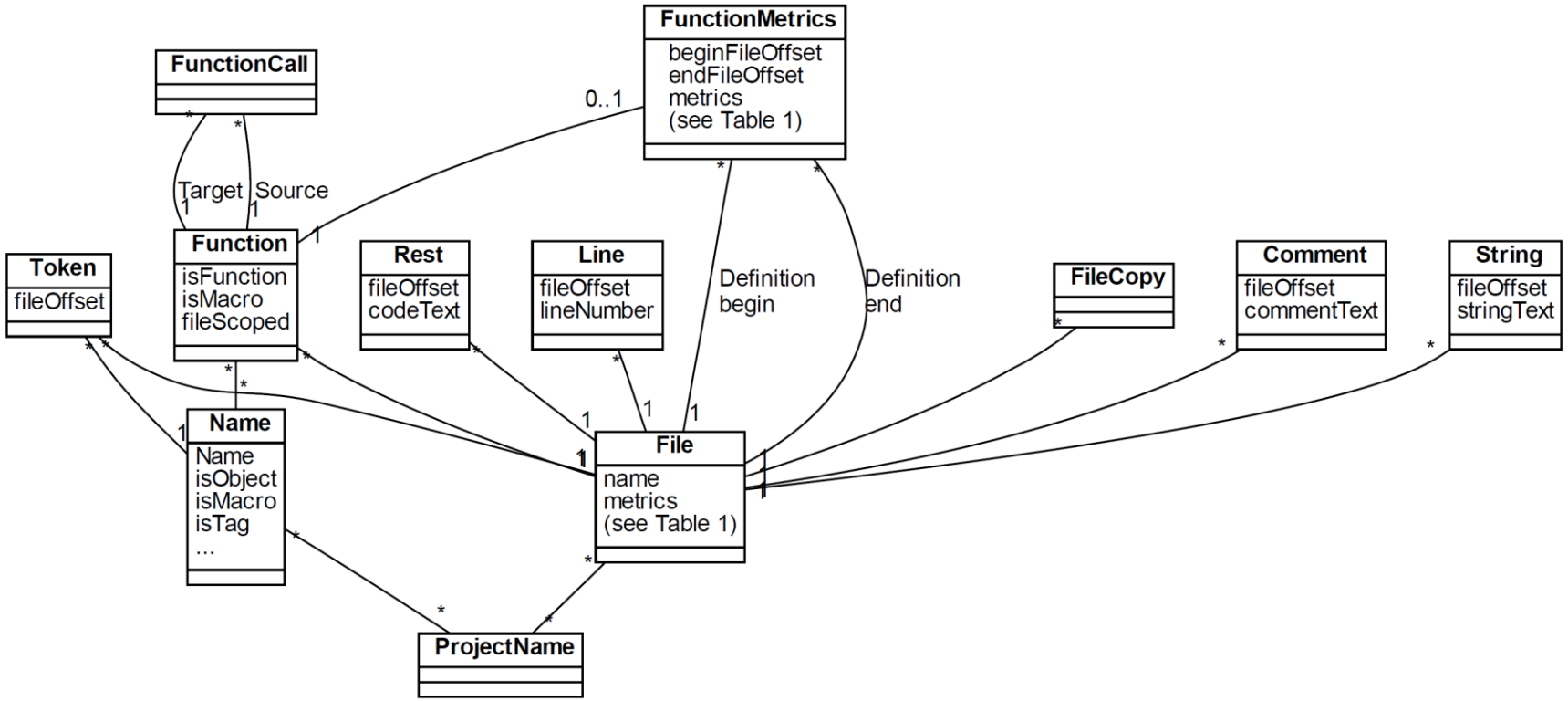
◉ Match any marked ○ Match all marked ○ Exclude marked ○ Exact match

Number of direct callers ignore
Function names should (not) match RE
Names of calling functions should (not) match RE
Names of called functions should (not) match RE
Select functions from filenames (not) matching RE

Query title [Show functions] [Show files]

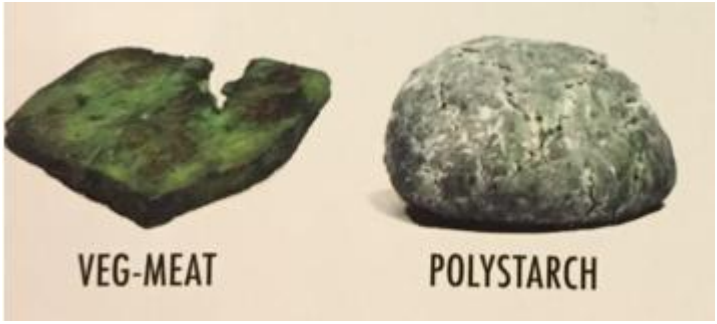
Main page — Web: Home Manual

Database Schema



Piecing together file 42

```
select s from (
 select name as s, foffset
 from ids inner join tokens on ids.eid = tokens.eid where fid = 42
 union select code as s, foffset from rest where fid = 42
 union select comment as s, foffset from comments where fid = 42
 union select string as s, foffset from strings where fid = 42
) order by foffset
```



Examples

	awk	Apache httpd	FreeBSD kernel	Linux kernel	Solaris kernel	WRK	PostgreSQL	GDB
Overview								
Configurations	1	1	3	1	3	2	1	1
Modules (linkage units)	1	3	1,224	1,563	561	3	92	4
Files	14	96	4,479	8,372	3,851	653	426	564
Lines (thousands)	6.6	59.9	2,599	4,150	3,000	829	578	363
Identifiers (thousands)	10.5	52.6	1,110	1,411	571	127	32	60
Defined functions	170	937	38,371	86,245	39,966	4,820	1,929	7,084
Defined macros	185	1,129	727,410	703,940	136,953	31,908	4,272	6,060
Preprocessor directives	376	6,641	415,710	262,004	173,570	35,246	13,236	20,101
C statements (thousands)	4.3	17.7	948	1,772	1,042	192	70	129
Refactoring opportunities								
Unused file-scoped identifiers	20	15	8,853	18,175	4,349	3,893	2,149	2,275
Unused project-scoped identifiers	8	8	1,403	1,767	4,459	2,628	2,537	939
Unused macros	4	412	649,825	602,723	75,433	25,948	1,763	2,542
Variables that could be made static	47	4	1,185	470	3,460	1,188	29	148
Functions that could be made static	10	4	1,971	1,996	5,152	3,294	133	69
Performance								
CPU time	0.81”	35”	3h 43’40”	7h 26’35”	1h 18’54”	58’53”	3’55”	11’13”
Lines / s	8,148	1,711	194	155	634	235	2,460	539
Required memory (MB)	21	71	3,707	4,807	1,827	582	463	376
Bytes / line	3,336	1,243	1,496	1,215	639	736	840	1,086

Thank you!

www.spinellis.gr/cscout/

dds@aueb.gr

www.spinellis.gr

@CoolSWEng