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

```c
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));
}
```
```c
#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));
}
```
Refactoring Strategy

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

```c
struct a {
    int a, b;
};
struct b {
    int a, c;
};
void f(int a)
{
    struct a sa;
    struct b sb
    const struct b *bp = &sb;
    a:
    sa.a = sb.a = a;
    while (--a) {
        const struct b *bp = &sb;
    }
    goto a;
}
```
Lexical Equivalence

```c
#define a b

f(void)
{
  int a;
  b = 42;
}

struct sa {
  int v;
  char c;
};

struct sb {
  int v;
  double d;
};
#define val(x) ((x)->v)

int
equality(struct sa *ap, struct sb *bp)
{
  return (val(ap) == val(bp));
}
```

```c
typedef unsigned int uint;

static int s;

#include "header.h"

static uint
function_a()
{
  return 2 * s;
}
```

```c
#include "header.h"

static uint
function_b()
{
  return ++s;
}
```
Partial Lexical Equivalence

```c
#define a(x) h_ ## x

f(void)
{
    int a(tailname);
    h_tailname = 42;
}
```
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
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

```c
Code Browsing
```
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 occurrence(s)
- Appears in project(s):
  - maketab
  - a.out
  - /home/dds/src/cscout/example/awk/a.out

- **Dependent files**
- **Associated functions**
- The identifier occurs (wholly or in part) in function name(s):
  1. [execute] — function page

- Substitute with: execute  

Main page — Web: Home Manual
Generalized Queries

- Names
- Files
- Functions
Identifier Query

- 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

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

### Reverse sort order
- Match any of the above
- Match all of the above

File names should (not) match RE

**Query title**

[Main page](#) — Web: [Home](#) [Manual](#)
Function Query
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

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

www.spinellis.gr/cscout/

dds@aueb.gr

www.spinellis.gr

@CoolSWEng