

Team Acacia

Michael W. Godfrey
Assistant Professor
University of Waterloo
migod@plg.uwaterloo.ca

Andrew Trevors
Graduate Student
University of Waterloo
adtrevor@uwaterloo.ca

What Is Acacia ??

- Acacia is the work of Emdem Gansner
- Based around the EDG front end for C++
- Consists of
 - CCia - command line extractor
 - ciao - form based X-windows querying tool
 - cdef and cref - command line querying
- Available for commercial and non-commercial use on a variety of Unix platforms

CCia – The Command Line Extractor

- Supports the following standard compiler flags
 - -D & -U for macros, -I for include files
- Before using CCia, must specify the name of the compiler through the cc environment variable
 - Must be available at run-time
 - Probes compiler for “special settings” (e.g. predefined macros, include files)

CCia – The Command Line Extractor

- Typical all-at-once extraction :
`ksh CCia [-D <macroName>] [-U <macroName>]
[-I <include>] *.cpp`
- Wrapped around call to ksh in order to take advantage of flexibility that ksh offers
- Result of a successful extraction is a .A file for each .c, .cc, .C, .cpp file and entity.db and relations.db

What Have We Done?

- Information about the major entities of the software system and the relationships between them.
- Information on entities at the “external declaration” level. Local variables and AST-level relationships are not modeled.
- Entities include global standalone (extern,static) :
 - variables
 - functions
 - files
 - macros

What Have We Done?

- types (class, enum, struct, union)
- sub-parts of types (class methods, member variables, enum values, struct sub-parts)

Entity Queries - cdef

- `ksh cdef -u <kind> <name> [attr=val ...]`
- -u flag means “give complete but unformatted output
- <kind> must be one of f(ile), fu(nction), m(acro), v(ariable), t(ype).
- <name> is the name name of the entity
- “-” can be used as a wildcard
- 17 columns of semi-colon delimited output

Relationship Queries - cref

- `ksh cref -u <kind1> <name1> <kind2> <name2>
[attr=val ...]`
- 42 columns of semi-colon delimited output
 - 18 columns for entity 1
 - 18 columns for entity 2
 - 4 columns of relationship attributes

Functions Test Bucket?

- In main(), an array of functions, fp, is defined. Where are these functions defined and how are they found from main.C?
- `ksh cdef -u fu squared
main.C;6;6;dec;extern
multiply.C;1;3;def;static
squared.C;1;3;def;extern`
- `ksh cref -u -- fu squared file1=main.C
squared.C;1;3;def;12;reference`

Functions Test Bucket?

- ksh cref -u - - fu sqrt file1=main.C
/usr/include/bits/mathcalls.h;146;146;dec;12;reference
- ksh cref -u - - fu tan file1=main.C
/usr/include/bits/mathcalls.h;67;67;dec;12;reference
- ksh cref -u - - fu fabs file1=main.C
/usr/include/bits/mathcalls.h;165;165;dec;12;reference

Functions Test Bucket?

- ksh cref -u - - fu triple file1=main.C
multiply.C;5;7;def;12;reference
- ksh cref -u - - fu whacked file1=main.C
multiply.C;13;16;def;12;reference

Functions Test Bucket?

- In multiply.C, the function whacked calls the squared function. Which function does this call resolve to?
Same way as question 1, resolution is correct.
- Where is the type ptr2func define? (Give your answer in bytes from start off file.)?
Acacia does not model at the byte-level
- Where are the calls to triple?
The explicit call to triple in whacked is caught, as is the reference (like a variable) in sort.C. The other uses are implicit and not caught.

How Did We Do Overall?

- We were able to answer a fair share of questions
- However, Acacia is not a robust parser
 - does not model local variables or AST-level relationships
 - all required “include” files must be made available
 - code must be compilable
 - embedded languages (SQL, asm, etc) are not supported