The Automated Analysis of Header Files for Support of the Standardization Process

Eugene Novikov
Institute for System Programming, RAS
joker@ispras.ru
Denis Silakov
Institute for System Programming, RAS
silakov@ispras.ru

Linux Standard Base (LSB)

- Large number of Linux distributions
  - Different versions and specific modifications of components
  - Absence of complete compatibility at the binary level
  - Complexity of application maintenance and porting
- LSB is aimed to help developers to create and support portable applications for Linux operating system

LSB database

- LSB database is the LSB core
- From LSB database it is generated
  - LSB standard text
  - Primitive tests
  - Environment for LSB-compatible applications development

LSB database basic objects

Libraries	~60
Header files	~900
Interfaces	~47000
Types	~16000
Macros	~12000
C++ classes	~1600

LSB database population

- Libraries binary files
  - With debugging information
  - Without debugging information
- Header files

Binary files analysis by *libtodb*

- Interfaces exported by libraries
- Binary symbols versions
- Absence of some data (e.g. inline functions, preprocessor directives, etc.)
- Insufficient C++ support
- Absence of interrelations between complex types

Header files analysis by *headertodb*

- Obtaining of considerable part of data from header files
- Insufficient analysis by *ctags* program
- Insufficient C++ support
- Absence of interrelations between preprocessor directives and other entities

Goals of this work

- Develop a method for header files analysis
  - Allow automated header files analysis
  - Provide necessary C++ support
  - Allow to interrelate complex entities with each other
  - Allow to interrelate preprocessor directives with other entities
- Develop a tool to implement this method

Suggested approach

- Based on *cpp* preprocessor and *gcc* compiler high-level representations
  - Analyzers supported by third-party developers
  - Simple and formal representation structure
  - Most detailed analysis of Linux header files
  - The openness of the *cpp* and *gcc* source code
  - Absence of the detailed documentation
  - Incompleteness of data
  - Possible changes in high-level representations

Gcc compiler parsing tree structure

- *Gcc* compiler parsing tree in text representation
  - Nodes corresponding to entities (`@731`)
  - Their attibutes
- The first attribute
  - Entity kind (`integer_type`)
- Subsequent attributes values
  - Reference to another node (`@2449`)
  - Some text information (`long double`)
  - Entity location in header file (`timer.h:241`)

Function declaration

- The first attribute is `function_decl`
- Subsequent attributes names and values
  - `name` – reference to function name identifier
  - `type` – reference to function signature
  - `srcp` – function declaration location in header file
  - `scpe` – reference to function scope, either class or namespace
  - `accs` (optional) – access to class method
  - `spec` (optional) – class method specifier
  - `note` (optional and multiple-valued) – constructor, destructor, operator, etc.

Function declaration extensions

- Extended attributes names and values
  - `ext_note` (optional and multiple-valued) – explicit, inline, throw
  - `ext_qual` (optional) – class method qualifier
  - `ext_body` (optional) – reference to expression corresponding to function body
  - `ext_body_open_brace` (optional) – opening brace location, the beginning of function body
  - `ext_body_close_brace` (optional) – closing brace location, the end of function body

Additional analyzers

- Preprocessor conditional compilation directives
  - Conditions (`#if`, `#ifdef` and `#ifndef`)
  - Branches (`#else` and `#elif`)
  - Conditional compilation end (`#endif`)
- Special comments
  - LSB parameters
  - LSB IDs

headertodb3 tool

- Input is header files
- Major tool work stages
  - Preprocessor directives and special comments analysis
  - Entities ordering
  - Ordered entities analysis by means of special handlers
- Output is SQL script

headertodb3 application in Qt4 library standardization process

	libQtCore	libQtGui
Header files	85	183
Interfaces	4450	9720
Classes	440	990
Enumerations	160	320
Templates	240	155
Typedefs	290	240
Macro definitions	290	190
Entities	22760	40310
Properties and interrelations	87830	161030

Conclusion

- It was developed the method that allows to analyze header files on the basis of *cpp* and *gcc* high-level representations
- *Headertodb3* tool provides both C and C++ support
- Tool makes header files analysis substantially automated
- *Headertodb3* was successfully applied during Qt3 and Qt4 libraries standardization process

Thank you!