Microsoft Visual C++ Demangling

Michael LeSane, Center for Computation & Technology, Louisiana State University

What are Mangled Names?

- Refer to strings generated by compilers containing encoded information about declarations such as functions, operator overloaders, templates, and static objects; also include details such as typing, arguments, and identifiers
- Used by the linker in order to derive information about such declarations, as well as to distinguish between declarations with identical names in different namespaces.
- Are more compact and parsing is easier for the compiler as a result

Types of Mangling Schemes

There are a number of different mangling schemes, but the most common types for the C++ programming language are those utilized by the GNU C++ Compiler and the Microsoft Visual C++ compiler.

The Main Issue

Whereas the mangling scheme of GCC, an open-source compiler, is well-documented and open-source de-mangling code in both the C and C++ programming languages is readily available, Microsoft offers little information regarding how to parse such strings.

To my knowledge, no open-source Visual C++ de-mangling code written in C++ has existed before now, with most existing code on Unix-like systems depending on C-based libraries developed by the GNU project, and code on Windows systems making direct calls to the win32 API.

My work this summer been to develop a C++-based de-mangling library for the HPX project.

Examples of Visual C++ Mangled Names

Declaration	Visual C++
int myint;	?myint@@3HA
int qualifier::mystaticint;	?mystaticint@qualifier@@3HA
int* myintptr;	?myintptr@@3PAHA
void* myvoidptr;	?myvoidptr@@3PAXA
int f_i_vptr(void*);	?f_i_vptr@@YAP6AHXA
void* func_vptr_i(int);	?f_i_vptrv@@YAPAHPAX@Z
int (*funcptr)(int);	?funcptr_i_i@@3P6AHH@ZA
int mytemplate<int>::method(void);	?method@?$mytemplate@H@@YAHXZ

Selected Parsing Grammar Examples

mangledname	'?' basicname qualification signature storageclass
basicname	ident '@' '?' operatorcode '?$' template
qualification	'@' ident '@' qualification
signature	fcode functionsignature dcode datasignature
functionsignature	callingconvention typecode typecode arguments
fcode	'Y' (Complex) 'S' (Static) 'U' (Virtual)
datasignature	typecode
dcode	'3' '2'
ident	(A-Z||a-z||_)+ (A-Z|a-z|_|0-9)*
template	ident '@' arguments '@'
arguments	typecode arguments typecode '@' 'X' (void) 'Z'

Selected Code Examples

typecode	'D' (char) 'H' (int) 'M' (float) Specialtypecode Backref
specialtypecode	Have specific grammar rules; are similar to function signatures; include pointers and references
Callingconvention	'A' (__cdecl) 'E' (__thiscall) 'G' (__stdcall)
Storageclass	'A' (normal) 'B' (const) 'C' (volatile)
operatorcode	'0' (constructor) '6' (operator<<) 'H' (operator+)
Backref	[0-9] (refer to previous arguments or basicname instances)

Tasks

My initial task was to search for already existing open-source code released under a compatible-license. What I did find was in another language, and virtually impossible follow the logic of, or to port without also porting additional libraries in a very large project.

As a result, I was tasked with writing a new code to carry out the task, and eventually developed a program capable of parsing the 70-80 examples provided by various resources.

Since then, I have been focusing on incorporating features from the most recent de-mangling scheme while attempting to parse the thousands of lengthy mangled names that are a part of HPX's compiled binaries and linked libraries on Windows.

Other Issues

Because there is no publicly available official documentation on how to de-mangle the Microsoft Visual C++ scheme, the only resources available are derived from reverse-engineering.

Of these resources, there is no single clear and comprehensive resource regarding the parsing grammar, and few offer complete details regarding the various code definitions and rules associated with them.

Open source de-mangling code which already exists, such as that of the Wine project, cannot parse the most recent Visual C++ mangling scheme.

Most open source de-mangling code is released under licenses incompatible with that of the HPX project. Code released under compatible licenses still must be re-licensed under that of HPX before being incorporated into the project's core libraries, for the sake of uniformity. In order to do this, however, the original author's permission is required.

Significance to HPX and Beyond

- Will help the HPX project by facilitating the establishment of a common format between mangled names of incompatible schemes
- Could be of use to the LLVM project / Clang compiler, used by Mac OS X and the FreeBSD project; while it is capable of mangling names according to the Visual C++ scheme, it contains no library for de-mangling them
- Could be of use to the ReactOS project, which aims to develop a free, open-source , and binary-compatible clone of Microsoft Windows