|  |  |  |
| --- | --- | --- |
| Syntax of SASL Symbol Name Mangling | September 26  2010 | |
| This script describes the rules of symbol name mangling. | | V1.0 |

SASL Name Mangling Syntax

V1.0

**Name mangling** can be applied to name of function mainly, for distinguishing the overloaded function

MangledName is the result of Name Mangling. It starts with ‘M’ character and ends with ‘Z’ character.

MangledName ::= ‘M’ QualifiedName ‘@@’ ParameterTypes CallingConvention

QualifiedName is base name qualified by scope name. A scope name could be a structure name, class name or namespace name.

ParameterTypes describes the types of parameter list.

‘@@’ is the splitter between qualified name and parameter type.

QualifiedName ::= ( ScopeName ‘@’ )\* BaseName

QualifiedName is composited by scope name and base name. ‘@’ is used to split the scope segments.

ScopeName ::=

‘S’ StructTypeName

‘N’ NamespaceName

StructTypeName ::= <name of structure>

*NamespaceName ::= <name of namespace>*

*BaseName ::= <literal name of function>*

ScopeName are two sorts, structure name or namespace name. Sorts are distinguished by a lead character. If the name starts with ‘S’ it means the name of structure, and ‘N’ means following name represents a namespace. Note that, if

BaseName is the literal name of the function or a type.

ParameterTypes is a list of parameter types. Because SASL do not support variance parameters, so we need not to considerate that. Syntax of ParameterTypes as following:

ParameterTypes ::= ( ParameterType ‘@@’)\*

ParameterType is encoded by type qualifiers and type name.

***NOTE THAT***, anywhere type name referred means that the original definition name but not alias( such as generated by “typedef” ).

ParameterType ::= ScopeQualifiedTypeName

ScopeQualifiedTypeName is the type name qualified by scopes. It is similar with qualified name.

ScopeQualifiedTypeName ::= ( ScopeName ‘@’ )\* BaseTypeName

ScopeName ::= ( ‘S’ StructTypeName ) | ( ‘N’ NamespaceName )

QualifiedBaseTypeName ::= TypeQualifier BaseTypeName

TypeQualifier ::= (‘C’)?(‘S’)?(‘U’)?Q

TypeQualifier describes the qualifier of declaration. In SASL, following qualifiers are supported: constant qualifier (‘C’), uniform qualifier (‘U’) and shared qualifier (‘S’). Each qualifier is represented as a character, and qualifiers is ended by a character ‘Q’.

BaseTypeName ::=

‘S’ StructTypeName

‘B’ BuildinScalarTypeName

‘V’ BuildinVectorTypeName

‘F’ FunctionTypeName

‘M’ BuildinMatrixTypeName

‘A’ ArraySize ArrayComponentTypeName

StructTypeName ::= <literal name of structure>

BuildinScalarTypeNames encode all build-in type into one or two characters.

BuildinScalarTypeName ::=

‘U1’ | ‘U2’ | ‘U4’ | ‘U8’ |‘I1’ | ‘I2’ | ‘I4’ | ‘I8’ | ‘B’ | ’F’ | ‘D’ | ‘O’

ArrayComponentTypeName ::= QualifiedBaseTypeName

ArraySize ::=

<length of array >

‘V’

They are corresponded following building types: ***uint8\_t***, ***uint16\_t***, ***uint32\_t***, ***uint64\_t***, ***int8\_t***, ***int16\_t***, ***int32\_t***, ***int64\_t***, ***boolean***, ***float***, ***double*** and ***omit*** (***void***).

BuildinVectorName ::= VectorLength BuildinScalarTypeName

BuildinMatrixName ::= RowCountOfMatrix ColumnCountOfMatrix BuildinScalarTypeName

BuildinVectorTypeName shows the name of building vector. It makes of length component and scalar type component.

VectorLength Could be 1, 2, 3 or 4 and represents the length of vector.

And the BuildinMatrixTypeName is similar as build-in vector type name.

ArraySize is an integer demonstrating the size of fixed-length array or ‘V’ (void or variant) means that it is a variant length array.

ArrayComponentTypeName is the description of type of component of array. For example, array component type of “int32\_t[]” is “int32\_t”. So array component type name is set to the mangled name of int32\_t, “I4”.

CallingConvention is a single character which shows the calling convention of this function. ‘S’ means “\_\_stdcall”, ‘C’ means “\_\_cdecl” and ‘F’ ,means “\_\_fastcall”

CallingConvention ::=

‘S’

‘C’

‘F’

FunctionTypeName reuses all non-terminators and terminators mentioned above.

FunctionTypeName ::= ReturnType PrarameterTypes CallConvetion ‘@@@’

ReturnType ::= ParameterType ‘@@’