C++: Namespaces

Miro Jurišić

meeroh@meeroh.org

What is a namespace?

- Namespace: a naming context
- Conflicts are bad
- Make names unique within a namespace, make namespaces unique
- A C++ namespace: contains classes, variables, constants, functions, etc.
- Names in a namespace are visible outside the namespace
- Example: a class is a namespace

```
class Example {
    public:
        void PublicFunction ();

    private:
        void PrivateFunction ();
};
```

- Both PublicFunction() and PrivateFunction() are visible outside the class, as Example::PublicFunction() and Example::PrivateFunction()

Why do we need namespaces?

- Need to distinguish between different items with the same name
- Example: two libraries both have a String class
- C solution: make the names different (Library1String and Library2String)
- Requires vendors to cooperate, and makes all names longer (kThemeWidgetCloseBox)
- Bad C++ solution: use dummy classes or structs to group names

```
struct Library1 {
    static      void    Function1 ();
};

struct Library2 {
    static      void    Function1 (int);
};
```

- Requires the entire library to be in one header file (or included from it)
- Everything not in a class or in a function has to be global

Namespace syntax

```
namespace MyNamespace {
    void Function1 ();
    void Function2 ();

    typedef UInt32      MyInt32;

    MyInt32         gVariable;
}
```

- Very similar to class syntax
- No access specification (public, private, protected)
- No trailing semicolon
- Namespaces are open – they can be in several independent header files

```
Library1String.h:

namespace Library1 {
    class String;
}

Library1List.h:

namespace Library1 {
    class List;
}
```

Using a namespace

```
namespace Library1 {
    class String;
    class List;
}
```

- Explicit qualification

```
void DoSomething ()
{
    Library1::String      string;
    Library1::List        list;

    // ...
}
```

- using declaration

```
void DoSomething ()
{
    using Library1::String;
    using Library1::List;

    String              string;
    List                list;

    // ...
}
```

- using directive

```
void DoSomething ()
{
    using namespace Library1;

    String              string;
    List                list;
}
```

Resolving conflicts

```
namespace Library1 {
    class String;
    class List;
}

namespace Library2 {
    class String;
    class List;
}
```

* Explicit qualification

```
void DoSomething ()
{
    Library1::String        string1;
    Library2::String        string2;
}
```

- using declaration

```
void DoSomething ()
{
    using Library1::String;

    // Really Library1::String
    String              string;
    Library2::String        string;
}
```

- using directive

```
void DoSomething ()
{
    using namespace Library2;
    using Library1::String;

    // Really Library1::String
    String              string;
    // Really Library2::List
    List                list;
```

Nested namespaces

```
namespace Library1 {
    namespace Part1 {
        class String;
        class List;
    }

    namespace Part2 {
        class String;
        class List;
        class Array;
    }
}
```

- Use any combination of access techniques

```
void DoSomething ()
{
    Library1::Part1::String          string1;
    Library1::Part2::String          string2;

    using Library1::Part1::String;
    String                           string3;

    using namespace Library1::Part1;

    // Really Library1::Part1::List
    List                             list1;

    using namespace Library1;

    // Really Library1::Part2::List
    Part2::List                      list2;
```

```
    // Really Library1::Part2
    using namespace Part2;
    // Really Library1::Part2::Array
    Array                           array;
}
```

- Namespace aliasing

```
void DoSomething ()
{
    namespace LP2 = Library::Part2;

    // Really Library::Part2::String
    LP2::String                 string;
}
```

The anonymous namespace

```
namespace {
    Boolean              gAllDone;
    ProcessSerialNumber  gMyPSN;
}
```

- The anonymous namespace is not accessible outside the compilation unit
- Replaces `static` for functions and variables shared within one source file
- Avoids using `static` to mean two completely different things

The `std` namespace

- `std` namespace contains entire C++ standard library
- Old-style headers (`vector.h`, `string.h`, etc) still put everything in the global namespace
- Better to use new-style headers (`vector`, `string`, etc)
- For C library, new-style headers start with "c" (`cstdio`, `cstdlib`, etc)

Köenig Lookup

- aka Argument-Dependent Lookup
- Sane rules for namespace lookup
- Helps avoid explicit qualification

```
// ...

result = function (argument1, argument2, ...);

// ...
```

- At a function call site, the name of function is looked up in the arguments' namespaces

```
// ...

myns::number x;
myns::number y;
```

```
myns::number z = max (x, y);

// ...
```

- `::max` and `myns::max` are both considered, and `myns::max` wins
- More specifically:
  - The current namespace and the global namespace are always searched
  - For each argument, additional namespaces are searched
  - All candidate namespaces are collated
  - Function name is looked up in the collated list
  - Overloads and conflict resolution proceed as normally

Namespaces in your code

- Can adopt them incrementally
- Use namespaces instead of long names
- You can always shorten the names if necessary with aliases and using directives and declarations

```
class PreferencesDialog
{
    namespace       MailPreferences {
        enum {
            Server          = 128,
            Username        = 129
        }
    }

    namespace       NewsPreferences {
        enum {
            Server          = 131,
            Username        = 132
```

```
            }
        }
        // ...
    };


    void PreferencesDialog::ItemHit ()
    {
        switch (itemHit) {
            case MailPreferences::Server:
            // ...

            case MailPreferences::Username:
            // ...

            case NewsPreferences::Server:
            // ...

            case NewsPreferences::Username:
            // ...
        }
    }
```

```
void PreferencesDialog::SetupMailPreferences ()
{
    using MailPreferences;

    SetValue (Server, server);
    SetValue (Username, username);
}
```