[ab]Using Enum

Елена Степанова
st_1ena
elena.1.stepanova@nokia.com

Packet Core developer
Product owner
Nokia

Enum evolution

Enum evolution

```
enum {
        TCP_LINK,
        UDP_LINK,
        SCTP_LINK,
        MAX_LINKS = SCTP_LINK + 1
};
```

Enum evolution

```
enum link_type {
      TCP_LINK,
      UDP_LINK,
      SCTP_LINK,
      MAX_LINKS = SCTP_LINK + 1
};
```

Enum evolution

```
enum class link_type {                                  // C++11
      TCP_LINK,
      UDP_LINK,
      SCTP_LINK,
      MAX_LINKS = SCTP_LINK + 1
};
```

Enum evolution

```
enum class link_type: unsigned char {                    // C++11
      TCP_LINK,
      UDP_LINK,
      SCTP_LINK,
      MAX_LINKS = SCTP_LINK + 1
};
```

Enum evolution

```
enum class link_type: unsigned char;                    // C++11

enum class link_type: unsigned char {                   // C++11
      TCP_LINK,
      UDP_LINK,
      SCTP_LINK,
      MAX_LINKS = SCTP_LINK + 1
};
```

Enum evolution

```
enum class link_type: unsigned char;                    // C++11

enum class link_type: unsigned char {                   // C++11
      TCP_LINK [[deprecated("Use SCTP instead")]],      // C++17
      UDP_LINK,
      SCTP_LINK,
      MAX_LINKS = SCTP_LINK + 1
};
```

Enum evolution

```
enum class link_type: unsigned char;                    // C++11

enum class link_type: unsigned char {                   // C++11
      TCP_LINK [[deprecated("Use SCTP instead")]],      // C++17
      UDP_LINK,
      SCTP_LINK,
      MAX_LINKS = SCTP_LINK + 1
};

using enum links;                                       // C++20
```

Syntax

Syntax

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

link_type scoped = link_type::TCP_LINK;
```

```cpp
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

link_type scoped = link_type::TCP_LINK;

using enum link_type;
link_type unscoped = UDP_LINK;
```

```cpp
// works as
using link_type::TCP_LINK;
using link_type::UDP_LINK;
using link_type::SCTP_LINK;
using link_type::MAX_LINKS;
```

Syntax

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

link_type scoped = link_type::TCP_LINK;

using enum link_type;
link_type unscoped = UDP_LINK;

using enum link_type::MAX_LINKS;
```

Use cases

Use cases: cozy switching

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };
// ...
std::string_view to_string(link_type link) {
      switch (link) {

            case link_type::TCP_LINK:   return "TCP link";
            case link_type::UDP_LINK:   return "UDP link";
            case link_type::SCTP_LINK:  return "SCTP link";
            default: return "Unsupported link";
      }
}
```

Use cases: cozy switching

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };
// ...
std::string_view to_string(link_type link) {
      switch (link) {
            using enum link_type;
            case TCP_LINK:  return "TCP link";
            case UDP_LINK:  return "UDP link";
            case SCTP_LINK: return "SCTP link";
            default:        return "Unsupported link";
      }
}
```

Use cases: import into namespace

```
namespace networking {
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

}
// ...
networking::link_type get_link_type() {
      // ...
      return networking::link_type::SCTP_LINK;
}
```

Use cases: import into namespace

```
namespace networking {
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };
using enum link_type; // imports enumerators into namespace
}
// ...
networking::link_type get_link_type() {
      using namespace networking;
      return SCTP_LINK; // “unscoped” AND strongly-typed
      // return 1; ← error: cannot convert 'int’
      //                    to 'networking::link_type’
}
```

Use cases: import into struct/class

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

class Links {
public:
      Links() = default;
      using enum link_type;
};

static_assert(link_type::TCP_LINK == Links().TCP_LINK);
static_assert(link_type::TCP_LINK == Links::TCP_LINK);
```

Use cases: import into struct/class

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

class Links {
public:
      Links() = default;
private:
      using enum link_type;
};
static_assert(link_type::TCP_LINK == Links().TCP_LINK);
// error: 'link_type Links::TCP_LINK' is private within this context
```

COMPILER EXPLORER

Add... More

C++ source #1

C++

```
1   #include <cassert>
2
3   enum class link_type: unsigned char
4   { TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1};
5
6   class Links {
7   private:
8       using enum link_type;
9   public:
10      Links() = default;
11  };
12
13  static_assert(link_type::TCP_LINK == Links().TCP_LINK);
14
```

x86-64 gcc 11.1 (C++, Editor #1, Compiler #1)

x86-64 gcc 11.1 -std=c++2a -Wall -Werror

Output... Filter... Libraries Add new... Add tool...

```
1   <No assembly to display (~77 lines filtered)>
```

Output (0/0) x86-64 gcc 11.1 - 1854ms (13258) ~77 lines filtered

x86-64 gcc 11.2 (C++, Editor #1, Compiler #2)

x86-64 gcc 11.2 -std=c++2a -Wall -Werror

Output... Filter... Libraries Add new... Add tool...

```
1   <Compilation failed>
2
3   # For more information see the output window
```

Output (0/6) x86-64 gcc 11.2 - 365ms

Output of x86-64 gcc 11.2 (Compiler #2)

Wrap lines

```
<source>:13:46: error: 'link_type Links::TCP_LINK' is private within this context
   13 | static_assert(link_type::TCP_LINK == Links().TCP_LINK);
      |                                              ^~~~~~~~
<source>:8:29: note: declared private here
    8 |         using enum link_type;
      |                             ^
Compiler returned: 1
```

Use cases: import into struct/class

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

class Links {
public:
      using enum link_type;
};

class Link_impl: public Links {};

static_assert(link_type::TCP_LINK == Link_impl().TCP_LINK);
```

Use cases: import into struct/class

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

class Links {
public:
      using enum link_type;
};

class Link_impl: public Links {};

static_assert(link_type::TCP_LINK == Link_impl().TCP_LINK);
```

```
// works as using-directive,
// no actual declaration
using link_type::TCP_LINK;
using link_type::UDP_LINK;
using link_type::SCTP_LINK;
using link_type::MAX_LINKS;
```

Pitfalls

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

enum class more_links { SCTP_LINK, DCCP_LINK, RDP_LINK };

using enum link_type;

using enum more_links; // error: 'more_links
more_links::SCTP_LINK' conflicts with a previous declaration
```

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

class Links {
      using enum link_type;
      using enum link_type;
      // error: redeclaration of 'using link_type::TCP_LINK’
      // error: redeclaration of 'using link_type::UDP_LINK’
      // error: redeclaration of 'using link_type::SCTP_LINK’
      // error: redeclaration of 'using link_type::MAX_LINKS'
};
```

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

using enum link_type;
using enum link_type; // Ok

namespace links {
using enum link_type;
using enum link_type; // Ok
}
```

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

struct Links {
      using enum link_type;
};

struct Link_impl: Links {
      using enum link_type; // Ok - shadowing
};
```

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };
enum class more_links { SCTP_LINK, DCCP_LINK, RDP_LINK };

struct Links {
      using enum link_type;
      using enum more_links; // error: redeclaration
};

struct Link_impl: Links {
      using enum more_links; // Ok - both are in place
};
```

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

struct Links {
      using enum link_type;
};

struct Link_impl: Links {
      static consexpr auto TCP_LINK = "42"; // Ok
};
```

```
#include <linux/netlink.h> // #define MAX_LINKS 32

enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };
//                               ^ error: expected identifier
```

```
enum class link_type: unsigned char
{ TCP_LINK, UDP_LINK, SCTP_LINK, MAX_LINKS = SCTP_LINK + 1 };

// ...

#include <linux/netlink.h> // MAX_LINKS became 32

using enum link_type;

// MAX_LINKS still 32
```

using enum:

\+ shorter, expressive code

— “invisible” constants

Спасибо за внимание!

Елена Степанова

st_1ena

elena.1.stepanova@nokia.com