C++ 标准库简介

分享人: 冯晓培

C Standards History

1970	Dennis Ritchie	C
1989	ANSI	ANSI C/ C89
1999	ISO	C99
2011	ISO	C11

C++ Standards history

1979	Bjarne Stroustrup	C++
1998	ISO	C++98
2003	ISO	C++03
2011	ISO	C++11
2014	ISO	C++14
2017	ISO	C++17
2020	ISO	C++20?

Bjarne Stroustrup’s blog: http://www.stroustrup.com

C++ std library

- std::string
- std::vector
- std::list
- std::stack
- std::queue
- std::set
- std::map
- std::multimap

string

```
void c_style()
{
	char str0[] = "jack";
	char str1[] = "@ikuai8.com”;
	char* dst = (char*)malloc(sizeof(str0) + sizeof(str1));
	memcpy(dst, str0, sizeof(str0));
	memcpy(dst+strlen(dst), str1, sizeof(str1));
	printf("c   style :%s\n", dst);
}

void cpp_style()
{
	string str0 = "jack";
	string str1 = "@ikuai8.com";
	string dst = str0 + str1;
	printf("cpp style :%s\n", dst.c_str());
}
```

string

- 常用操作：
- 初始化，访问，大小，修改，查找

- 示例代码：string_example.cc

vector

- dynamically sized array in **C**

```
typedef struct {
  int  size; // slots used so far
  int  capacity; // total available slots
  int  *data; // array of integers we're storing
} Vector;
```

- vector is build-in in c++

  示例代码：vector-example.cc

vector

- 特点：
- 内存动态增长，不主动收缩，gcc内存增长特征：1,2,3,4,5,6,7,8,16,32
- 操作复杂度

– Random access	O(1)
– Insert or remove at the end	O(1)
– Insert or remove of element	O(n)

stack and queue

- stack
  - first in last out
- queue
  - first in first out

示例代码：stack-example.cc queue-example.cc

set and map

- set
  - 排序
  - 唯一性

  set [1,2,3,4,5,6]
- map
  - 排序
  - 唯一性

  map {1:’a’} ,{2:’b’},{3,’c’}

  示例代码：set-example.cc map-example.cc

RAII

- RAII (Resource Acquisition Is Initialization)
- Type
- Object
- Lifetime
- Resource

problem

```
void fun(Mutex& mutex)
{
  mutex.Acquire();

  // do stuff here

  if (earlyOut)
  {
    // good thing I remember to do this
    mutex.Release();
    return;
  }

  // do stuff here

  mutex.Release();
}
```

```
void fun(Mutex &mutex)
{
  mutex.Acquire();

  // many lines of code

  if (newEarlyOut)
  {
    // oops...
    return;
  }

  // many lines of code

  mutex.Release();
}
```

C++ solution

```
class AutoMutexLock
{
  public:

    AutoMutexLock(Mutex &mutex)
      : m_mutex(mutex)
    {
      m_mutex.Acquire();
    }

    ~AutoMutexLock(void)
    {
      m_mutex.Release();
    }

  private:

    Mutex &m_mutex;
}
```

```
void fun(Mutex &mutex)
{
  AutoMutexLock lock(mutex);

  // many lines of code

  if (newEarlyOut)
  {
    // no need to release mutex here
    return;
  }

  // many lines of code

  if (earlyOut)
  {
    // nor here
    return;
  }

  // many lines of code

  // not at the end, either
}
```

c solution

- C with clang and gcc “cleanup” extension

```
static inline void fclosep(FILE **fp) { if (*fp) fclose(*fp); }
#define _cleanup_fclose_ __attribute__((cleanup(fclosep)))

void example_usage()
{
    _cleanup_fclose_ FILE *logfile = fopen("logfile.txt", "w+");
    fputs("hello logfile!", logfile);
}
```

RAII: resource

lock, file, socket, memory, db connection and anything that exists in limited suppy.

本节结束

- 求贤若渴
- 欢迎Linux C/C++开发 一起学习进步
- 内核、应用、服务端、嵌入式我们都要
- xpfeng@ikuai8.com