最新日志

ace框架与平台选择
IP地址网段1
IP地址网段
学习英语网站
Ubuntu 8.04
ACE写DLL工厂
安全专家建议的linux安装方法
linux版本
c++资源之不完全导引 （全文）__转
ACE配置文件
全部日志>>

最新评论

网络编程：受益良多!我的博客也是关于MySql方面的,www.sd...
网络编程：好文章，留个签名。我的博客也是关于ruby方面的,www...
AAA：抄袭他人文章，不注明出处，还写转载版权，实在无耻~~~ ...
网络编程：主题特出，好文章。我的博客也是关于watir方面的,ww...
Zero1："首先通过create_epoll(int m...
Jekinla：你好^_^,看似你对ACE还是有研究的,能不能帮我解决一...
Jekinla：你好^_^,看似你对ACE还是有研究的,能不能帮我解决一...
sss：有用户环境变量，还是系统环境环量...
codespy：我用VC2008,也编译了好几次4.4.0的商业版，系统...
vivien：博主这篇文章写的太好了，每次我装QT都会看。非常感谢分享...

2008-04-23 **ACE配置文件 - [ACE]**

ACE 是一个很大的东西，每个人学习路径可能都不一样。我的方法首先阅读 **C++ Network Programming, Volume 1** ，让自己明了**ACE** 的设计思路；再从实际的小项目入手，逐步的用 **ACE** 内容替换掉自己的东西，在比较和实践之间求得较深的了解。

就一个正常项目而言，一个配置文件是必不可少的，那就先从这里入手了。**linux/unix** 程序可能经常用到命令行方式，不过我还是比较喜欢 **windows** 的 **ini** 格式的，当然，有**xml** 的更好，不过 **ACE** 里暂时没有提供。配置文件的使用很简单，**ACE** 提供的类也很友好。在这里主要是整理一些细节，让大家不在这里浪费太多时间。

1、ACE 配置类简介与使用

先给出一个印象（为了直指主题，所有错误处理都被清除，具体例子请看最后源码）：

```
ACE_Configuration_Heap config;
 config.open();
ACE_Registry_ImpExp impExp(config);     // win32的注册表导出格式
//ACE_Ini_ImpExp impExp(config);          // windows的.ini格式，注意读出来的都是字符串，类型需要自己转换
impExp.import_config(ACE_TEXT("配置文件名"));     // 读取配置文件

ACE_Configuration_Section_Key section;// 指向一个配置节section
config.open_section(config.root_section(), ACE_TEXT("节名"), 0, section);

// 读取字符串
ACE_TString str;
config.get_string_value(section, ACE_TEXT("键名"), str); // ACE_Ini_ImpExp，只能使用这一种方法

// 读取整型
```


```
    u_int i;
    config.get_integer_value(section, ACE_TEXT("键名"), i);

   // 读取二进制
    void * data_out = 0; // 最好使用智能指针
    size_t length = 0;
    config.get_binary_value(section, ACE_TEXT("键名"), data_out, length);
    delete []data_out;
```

可以浏览一下 ACE_Configuration_Heap 和 ACE_Ini_ImpExp 或 ACE_Registry_ImpExp 之间的关系。不需要套上术语来理解，简化一下代码对此很有帮助。

一个超简化的ACE_Configuration_Heap

```
1  class ACE_Export ACE_Configuration_Heap : public ACE_Configuration
2  {
3    // 其它函数全部略去
4   private:
5     SECTION_MAP *index_;
6  };
```

值得注意的是 ACE_Registry_ImpExp 和 ACE_Ini_ImpExp ，这两个类都继承自 ACE_Config_ImpExp_Base，分别处理不同的格式。

简化版本的 ACE_Config_ImpExp_Base， 具体实现请看 Configuration_Import_Export.h

```
1  class ACE_Config_ImpExp_Base
2  {
3   public:
4    ACE_Config_ImpExp_Base (ACE_Configuration& config);
5    virtual int import_config (const ACE_TCHAR* filename) = 0;  // 从文件中导入配置
6    virtual int export_config (const ACE_TCHAR* filename) = 0;  // 导出配置到文件
7   protected:
8    ACE_Configuration &config_;
9  };
```

从两个简化的类，可以想象 ACE_Registry_ImpExp 和 ACE_Ini_ImpExp对import_config的具体实现会将不同格式的文件里的配置数据转化成相同的格式化的数据，并将之放到 ACE_Configuration_Heap 的 SECTION_MAP *index_。 ACE_Config_ImpExp_Base 具体实现的作用就是作为一个适配器。

2、ACE 配置格式说明：

（内容来自Configuration_Import_Export.h的注释）

2.1. ACE_Registry_ImpExp

使用win32 注册表文件导出文件格式，格式如下:

```
[Section]
    "key"="String Data"
    "key"=dword: 十六进制数值
    "key"=hex: 二进制
    注意，二进制的写法如下
    "pwd"=hex:31,32,33,34,35,36,37,38
```

2.2. ACE_Ini_ImpExp

从文件中导入字符串配置数据。充许 not-typed 值（没有 #，dword: hex:，等前缀），在标准的.ini和.conf中跳过空白字符(tabs和spaces)。 值（在等式的右边）可以被双引号限定，使得tabs或spaces能够留在字符串中。调用者必须自己转换字符串到类型。格式如下:

```
TimeToLive    =    100
Delay         =    false
Flags         =    FF34   # 可以象这样写注释
Heading       =    "ACE - Adaptive Communication Environment"
```

3、一个友好的函数

使用此类配置文件时，如果用到会变化的配置集合，我们经常遇到这样的写法

```
[section]
"size"=dword:2
"item_1"="value1"
"item_2"="value2"
```

这样写因为 **window** 提供的 **API** 只能一项一项读取，我们只能在读取了size 的值后，才能拼凑出"item_1"键名来取到相应的值。在实际操作中维护人员很容易遗漏操作，增加了选项但忘记增加size的值。ACE则提供了非常友好的enumerate_values函数，配合一下 **boost** 的 **split** 使用起来感觉很好。

一个非常简单的TCP转发程序的配置文件，将一端口接收到的数据转发至指定地址的端口，可以开多个服务端口:

```
[server\listen_items]
;listen_port, target_ip, target_port, timeout (second), validate
"item_1"="19998,192.168.168.217,8101,60,1"
"item_2"="8000, 192.168.0.57, 23, 60, 1"
```

实际的代码摘录，但不一定能编译呵 :-)

```
1
2  #include <boost/algorithm/string.hpp>
3  #include <boost/lexical_cast.hpp>
4  using namespace boost;
5
6
7  class ListenItem
8  {
9  public:
10   unsigned int listen_port_;
11   string target_ip_;
12   unsigned int target_port_;
13   unsigned int timeout_;
14   bool validate_;
15 };
16
17  vector<ListenItem> vec;
18
19  int index = -1;
20  ACE_TString name;
21  ACE_Configuration::VALUETYPE type;
22  while(0 == config.enumerate_values(section, ++index, name, type))
23  {
24   ACE_TString str;
25   if(config.get_string_value(section, name.c_str(), str) == -1)
26   {
27    ACE_ERROR_RETURN((LM_ERROR, ACE_TEXT("%p\n"), ACE_TEXT("ListenP
ort item does not exist\n")), -1);
28   }
29
30   ListenItem item;
31   vector<string> splitVec;
32   string ss = str.c_str();
33   split( splitVec, ss, is_any_of(",") );
34   item.listen_port_ = lexical_cast<unsigned int>(trim_copy(splitVec[0]));
35   item.target_ip_ = lexical_cast<string>(trim_copy(splitVec[1]));
36   item.target_port_ = lexical_cast<unsigned int>(trim_copy(splitVec[2]));
37   item.timeout_ = lexical_cast<unsigned int>(trim_copy(splitVec[3]));
38   item.validate_ =  lexical_cast<bool>(trim_copy(splitVec[4]));
39   if(item.validate_)
40    vec.push_back(item);
```

```
41 |
42 └ }
43
44
45
```

这样就可以不写

```
1  stringstream s;
2   s << "item_" << i+1;
3
```

这样的代码了，也避免了维护人员遗漏操作

4、编译选项

上面代码有用到 **boost** 的相关内容，如果你使用的是 ACE Programmer's Guide 的 2.5 How to Build Your Applications 节所介绍的Makefile，在 **linux** 下面的话是不可少的

CPPFLAGS = -i"$(BOOST_ROOT)"

当然，还要在.bash_profile或相应原地方加入类似说明:

BOOST_ROOT=$HOME/boost_X_XX_X
export BOOST_ROOT

//WuErPing 补充 (2006/12/28)

5、Use Unicode Character Set/Use Multi-Byte Character Set

上面的代码是使用Use Multi-Byte Character Set选项编译的，相应的编译ACE时也没有用到#define ACE_USES_WCHAR 1，如果使用Use Unicode Character Set是有问题的。要使写好的代码能同时在两个编译项中切换，除了利用ACE宏定义的ACE_TEXT，ACE_TString来处理字符串，用到的C++标准库string与stringstream时也需要做些处理。

```
1  #ifndef _SIDLE_APP_CONFIG_
2  #define _SIDLE_APP_CONFIG_
3
4  #include <vector>
5  #include <string>
6  #include <sstream>
7  #include <boost/algorithm/string.hpp>
8  #include <boost/lexical_cast.hpp>
9  #include "ace/Configuration_Import_Export.h"
10 using namespace std;
```

```
11  using namespace boost;
12
13  namespace TempfileManager
14{
15      typedef basic_string<ACE_TCHAR, char_traits<ACE_TCHAR>,
16          allocator<ACE_TCHAR> > stdstring;
17      typedef basic_stringstream<ACE_TCHAR, char_traits<ACE_TCHAR>,
18          allocator<ACE_TCHAR> > stdstringstream;
19
20      struct WorkInfo
21      {
22          stdstring name;    // 命名
23          unsigned long interval;    // 定时器间隔时间(秒)
24          unsigned long timeout;    // 定时器超时(秒)
25          bool forceable;    // 强制改变文件属性到普通
26          bool recursiveable;    // 递归处理子目录及下面的文件
27          unsigned int hour;        // 多少小时前的文件被处理 (小时)
28          unsigned int num;        // 每次最多处理文件数
29          stdstring directory;        // 目录
30          stdstring pattern;        // 通配符
31          bool workable;    // 是否交给定时器队列运行
32      };
33
34      class AppConfig
35
```


Tag：ace 配置文件 ACE_Configuration_Heap ACE_Registry_ImpExp ACE_Configuration_Section_Key

朋友， 我遇到一个难题，就是SVC config ,不能装入动态库。象我的/opt/ACE_wrappers/tests/DLL_Test

/opt/ACE_wrappers/examples/DLL 都不能生成动态库，您能告诉我如果在ACE下生成动态库吗？用它的例子为何不行呢？ 多谢了。 我QQ784838851

> **egeho123** 回复 **toby** 说:
>
> 你的开发环境是什么？可能原因是项目配置有问题
>
> 项目属性->常规->项目类型=动态库(.dll)；看是否是这个配置，如果还有问题，我错误贴出来吧
>
> 2008-04-30 09:08:48

toby | 发表于2008-04-27 22:13:11 [回复]

发表评论

用户名

匿名评论

密 码

提交