附录：工具类

地址包装类

ACE_INET_Addr

ACE_INET_Addr类封装了Internet域地址族（AF_INET）。该类派生自ACE中的ACE_Addr。它的多种构造器可用于初始化对象，使其具有特定的IP地址和端口。除此而外，该类还具有若干set和get方法，并且重载了比较操作，也就是==操作符和!=操作符。进一步的使用细节见参考文献。

ACE_UNIX_Addr

ACE_UNIX_Addr类封装了UNIX域地址族（AF_UNIX），它也派生自ACE_Addr。该类的功能与ACE_INET_Addr类相似。进一步的细节见参考文献。

时间包装类

ACE_Time_Value

通过ACE_DEBUG和ACE_ERROR记录日志

ACE_DEBUG和ACE_ERROR宏用于打印调试及错误信息及记录相应的日志。它们的用法已在整个教程中作了演示。

这两个宏可以使用下面的格式修饰符。就如在printf格式串中一样，这些选项的每一个都冠有前缀’%’：

格式修饰符	描述
‘a’	在该点结束程序（var参数是结束状态！）
‘c’	打印一个字符
‘i’，‘d’	打印一个十进制数
‘I’	根据嵌套深度缩进

‘e’，‘E’，‘f’，‘F’，‘g’，‘G’	打印一个双精度数
‘l’	打印错误发生处的行号
‘N’	打印错误发生处的文件名
‘n’	打印程序名（或“<unknown>”，如果没有设置的话）
‘o’	作为八进制数打印
‘P’	打印出当前的进程id
‘p’	从sys_errlist中打印出适当的errno值
‘r’	调用相应参数所指向的函数
‘R’	打印返回状态
‘S’	根据var参数打印出适当的_sys_siglist条目
‘s’	打印出一个字符串
‘T’	以hour:minute:sec:usec格式打印时间戳
‘D’	以month/day/year hour:minute:sec:usec格式打印时间戳
‘t’	打印线程id（如果是单线程则为1）
‘u’	作为无符号整数打印
‘X’，‘x’	作为十六进制数打印
‘%’	打印出一个百分号，’ %’

获取命令行参数

ACE_Get_opt

这个工具类基于stdlib中的getopt()<3c>函数，用于从用户那里获取参数。传入该类的构造器的名为optstring的串指定应用想要响应的“转换子”（switch）。如果“转换子”字母紧跟着一个冒号，那就意味着该“转换子”期望一个参数。例如，如果optstring为“ab:c”，应用就期望没有参数的“-a”和“-c”，和有一个参数的“-b”。例如，可以这样来运行此应用：

```
MyApplication -a -b 10 -c
```

()操作符已经被重载，并被用于扫描argv的成员，以找到选项串（optstring）中指定的选项。

下面的例子将帮助阐明怎样使用这个类来从用户那里获取参数。

```
#include "ace/Get_Opt.h"
int main (int argc, char *argv[])
{
//Specify option string so that switches b, d, f and h all expect
//arguments. Switches a, c, e and g expect no arguments.
ACE_Get_Opt get_opt (argc, argv, "ab:cd:ef:gh:");
int c;

//Process the scanned options with the help of the overloaded ()
//operator.
while ((c = get_opt ()) != EOF)
      switch (c)
      {
      case 'a':
            ACE_DEBUG ((LM_DEBUG, "got a\n"));
            break;

      case 'b':
            ACE_DEBUG ((LM_DEBUG, "got b with arg %s\n",
                        get_opt.optarg));
            break;

      case 'c':
            ACE_DEBUG ((LM_DEBUG, "got c\n"));
            break;

      case 'd':
            ACE_DEBUG ((LM_DEBUG, "got d with arg %s\n",
                        get_opt.optarg));
            break;

      case 'e':
```

```
            ACE_DEBUG ((LM_DEBUG, "got e\n"));
            break;

            case 'f':
            ACE_DEBUG ((LM_DEBUG, "got f with arg %s\n",
                        get_opt.optarg));
            break;

      case ’g’:
            ACE_DEBUG ((LM_DEBUG, "got g\n"));
            break;

      case ’h’:
            ACE_DEBUG ((LM_DEBUG, "got h with arg %s\n",
                        get_opt.optarg));
            break;

      default:
            ACE_DEBUG ((LM_DEBUG, "got %c, which is unrecognized!\n",c));
            break;
      }

//optind indicates how much of argv has been scanned so far, while
//get_opt hasn’t returned EOF. In this case it indicates the index in
//argv from where the option switches have been fully recognized and the
//remaining elements must be scanned by the called himself.
for (int i = get_opt.optind; i < argc; i++)
      ACE_DEBUG ((LM_DEBUG, "optind = %d, argv[optind] = %s\n",
                        i, argv[i]));

return 0;
}
```

有关如何使用此工具包装类的更多信息，请参见参考文献。

ACE_Arg_Shifter

这个ADT（抽象数据类型）将已知的参数或选项移动到argv向量的后面，这样在进行更深层次的参数解析时，就可以在argv向量的开始处定位还没有处理的参数。

ACE_Arg_Shifter将argv向量的各个指针拷贝到临时数组中。当ACE_Arg_Shifter在临时数组中遍历时，它将已知参数放至argv的尾部，未知的放至首部。这样，在访问了临时向量中的所有参数后，ACE_Arg_Shifter就以所有未知参数的初始顺序把它们放到了argv的前面。

这个类对解析来自命令行的选项也很有用。下面的例子将帮助演示这一点：

```
#include "ace/Arg_Shifter.h"


int main(int argc, char *argv[])
{
ACE_Arg_Shifter arg(argc,argv);
while(arg.is_anything_left ())
{
      char *current_arg=arg.get_current();
      if(ACE_OS::strcmp(current_arg,"-region")==0)
      {
            arg.consume_arg();
            ACE_OS::printf("<region>= %s \n",arg.get_current());
      }
      else if(ACE_OS::strcmp(current_arg,"-tag")==0)
      {
            arg.consume_arg();
            ACE_OS::printf("<tag>= %s \n",arg.get_current());
      }
      else if(ACE_OS::strcmp(current_arg,"-view_uuid")==0)
      {
            arg.consume_arg();
            ACE_OS::printf("<view_uuid>=%s\n",arg.get_current());
      }


      arg.consume_arg();
}


for(int i=0;argv[i]!=NULL;i++)
      ACE_DEBUG((LM_DEBUG,”Resultant vector”: %s \n",argv[i]));
}
```

如果这样来运行上面的例子：

```
.../tests<330> arg -region missouri -tag 10 -view_uuid syyid -teacher schmidt -student
tim
```

所获得的结果为：

```
<region> missouri
<tag>= 10
<view_uuid>=syyid
Resultant Vector: tim
Resultant Vector: -student
Resultant Vector: schmidt
Resultant Vector: -teacher
Resultant Vector: syyid
Resultant Vector: -view_uuid
Resultant Vector: 10
Resultant Vector: -tag
Resultant Vector: missouri
Resultant Vector: -region
Resultant Vector: ./arg
```