ACE/The ACE Programmer's Guide

@于龙

目录

[1 ACE简介 2](#_Toc418808278)

[1.1 ACE的组织结构 2](#_Toc418808279)

[1.2 消除C++编译器之间的差异 2](#_Toc418808280)

[2 使用ACE日志设置 3](#_Toc418808281)

[2.1 基本的日志记录和跟踪 3](#_Toc418808282)

[2.2 启用和禁用日志严重级别 3](#_Toc418808283)

[2.3 重定向日志输出 3](#_Toc418808284)

[3 收集运行时信息 3](#_Toc418808285)

[3.1 命令行参数和ACE\_Get\_Opt 3](#_Toc418808286)

[3.2 访问配置信息 4](#_Toc418808287)

[4 基本的TCP/IP Socket 用法 4](#_Toc418808288)

[4.1 一个简单的客户 4](#_Toc418808289)

[4.2 给客户增加健壮性 4](#_Toc418808290)

[5 处理事件及多个I/O流 4](#_Toc418808291)

[5.1 Reactor框架总览 4](#_Toc418808292)

[5.2 处理多个I/O源 4](#_Toc418808293)

# ACE简介

## ACE的组织结构

OS适配层：提供跨平台包装函数；

wrapper façade层：类型安全、面向对象的方式封装函数和数据；

框架层：提供可供复用的架构；

网络化服务层：完整的、可复用的服务；

## 消除C++编译器之间的差异

处理编译器差异的四个主要方面：

1. 模版；
2. 数据类型；
3. 运行时初始化和关闭；
4. 分配堆内存；

运行时初始化和关闭：

ACE通过以下三个相关的类提供了可移植的解决方案：

ACE\_Object\_Manager：管理对象，可以向其登记必须销毁的对象，在关闭时会以与登记次序相反的次序销毁所有已登记的对象；

ACE\_Cleanup：ACE\_Object\_Manager使用这个类的接口管理对象生命周期.

ACE\_Singleton：实现单例模式，相比全局对象其实例化和删除都在程序控制下，还增加了线程安全性，确保只有一个对象进行初始化.

必须遵循以下两条：

1. 永远不要直接调用exit()；
2. 确保ACE\_Object\_Manager成功初始化（在不使用标准main函数时）.

分配堆内存：

统一编译器在内存分配失败时返回0或抛出异常的行为，使用宏统一返回0.

窄字符和宽字符：

ACE\_HAS\_WCHAR：启用ACE的宽字符配置

ACE\_USES\_WCHAR：指示ACE在内部使用宽字符配置

ACE\_TCHAR：根据ACE\_USES\_WCHAR使用char或者wchar\_t

ACE\_TEXT：根据ACE\_USES\_WCHAR正确地定义字符串文字

字符串类：

ACE\_CString

ACE\_WString

ACE\_TString

# 使用ACE日志设置

## 基本的日志记录和跟踪

诊断输出宏：

ACE\_DEBUG

ACE\_ERROR

ACE\_TRACE

常用严重等级：LM\_DEBUG、LM\_ERROR

ACE\_TRACE会在自己所在的地方打印一行调试信息，并在退出其作用域时打印另外一行信息.

ACE\_ASSERT

## 启用和禁用日志严重级别

ACE为每个新派生的线程自动维护该线程专有的ACE\_Log\_Msg类的单体实例,ACE\_LOG\_MSG可以快捷访问.

日志优先级分为线程级和进程级.

任意级别启用了该消息的严重级别，消息就会记入日志.

## 重定向日志输出

ACE日志默认输出到标准错误流STDERR.

可以重定向到系统日志记录器SYSLOG.

# 收集运行时信息

## 命令行参数和ACE\_Get\_Opt

ACE\_Get\_Opt可以解析两种选项：

-开头的单字符

--开头的多字符

使用：

定义短选项；

迭代；

## 访问配置信息

ACE\_Configuration\_Heap在内存中保存所有信息；

ACE\_Configuration\_Win32Registry注册表API封装

ACE\_Registry\_ImpExp含有类型信息

ACE\_Ini\_ImmpExp Windows INT格式，没有类型信息.

# 基本的TCP/IP Socket 用法

## 一个简单的客户

ACE\_INET\_Addr 基类 ACE\_Addr.

ACE\_SOCK\_Stream对象表示已连接的TCP Socket.

## 给客户增加健壮性

检查ACE\_OS::last\_error()

# 处理事件及多个I/O流

## Reactor框架总览

使用Reactor框架，应用要实现其事件处理只需做三件事：

1. 从ACE\_Event\_Handler派生一个或多个类，并实现各个虚回调方法；
2. 向ACE\_Reactor登记事件处理对象，并与感兴趣的事件关联起来；
3. 运行ACE\_Reactor事件循环.

反应器模式，基础是事件多路分离.

## 处理多个I/O源

Reactor框架的常见用途是处理来自多个来源的I/O.

反应器会把一个ACE\_Event\_Handler指针与一个句柄以及该事件处理器感兴趣的I/O事件关联在一起.

get\_handler()获得句柄的挂钩方法.

挂钩方法返回-1，handle\_close就会被调用.