Module06-02
C++ ACE: Reactor 框架

C++ ACE - Reactor 框架

- ACE 简介
- I/O 相关对象
- ➔ Reactor 框架
- Service Configuration 框架
- Task 框架
- Acceptor-Connector 框架
- Proactor 框架
- 杂项

Reactor 框架 - 概要

- Reactor 框架：
  - 概要
  - ACE_Time_Value
  - ACE_Event_Handler
  - ACE_Timer_Queue
  - ACE_Reactor
  - Reactor 框架示例
  - Reactor 实现

Reactor 框架 - 概要

- 关于 ACE Reactor 框架
  - ACE Reactor 框架简化事件驱动程序的开发，而事件驱动是很多网络化应用的基本特征，这些应用常见的事件源包括 I/O 事件、 Posix 信号或 Windows 句柄激发以及定时器到期等
  - ACE Reactor 框架负责：
    - 检测来自各种事件源的发生
    - 将事件多路分离给其它事先登记的事件处理器
    - 分派各处理器所定义的挂钩方法，调用应用定义的逻辑处理这些事件

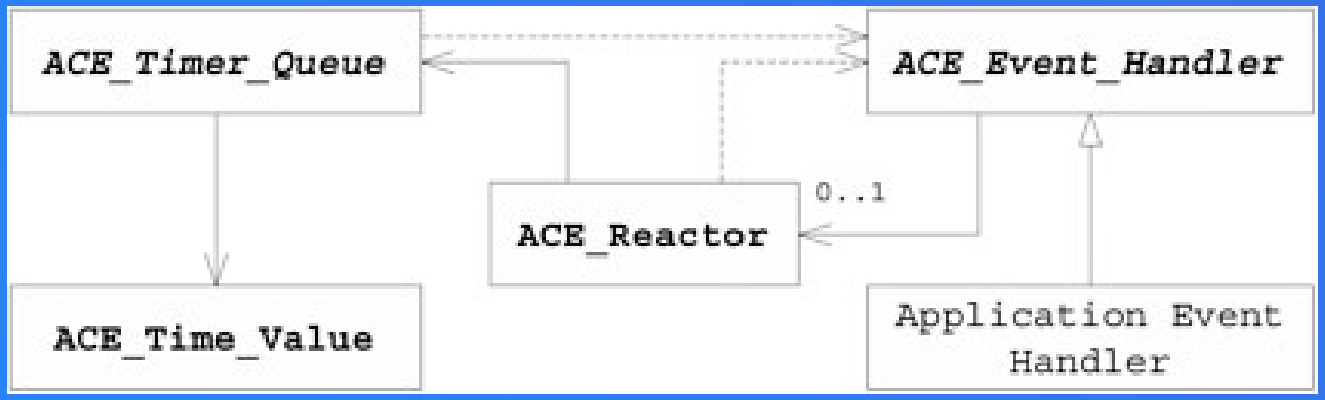
Reactor 框架 - 概要

- ACE Reactor 框架的主要参与者

ACE 类	说明
ACE_Time_Value	提供时间点和持续时间 (duration) 的可移植性、规范化的表示，使用 C++ 操作符重载来简化与时间有关的算术和关系运算
ACE_Event_Handler	抽象类，该接口定义的挂钩方法是 Reactor 回调的目标。应用程序提供的事件处理器为该类的派生类
ACE_Time_Queue	抽象类， ACE 有多种派生于 ACE_Time_Queue 的类，为不同的定时机制提供了灵活的支持
ACE_Reactor	提供一个接口，用来在 Reactor 框架中管理事件处理器登记，并执行事件循环来驱动事件检测、多路分离和分派

Reactor 框架 - 概要

- ACE Reactor 框架 类关系图



Reactor 框架 - ACE_Time_Value

- Reactor 框架：
  - 概要
  - ACE_Time_Value
  - ACE_Event_Handler
  - ACE_Timer_Queue
  - ACE_Reactor
  - Reactor 框架示例
  - Reactor 实现

Reactor 框架 - ACE_Time_Value

- 关于 ACE_Time_Value
  - ACE_Time_Value 类具有以下能力：
    - 提供了一种标准的时间表示，可以在各种操作系统上使用
    - 在不同的平台时间表示之间转换，如 Unix 上的 timespace_t 和 timeval ，以及 Windows 的 FILETIME 和 timeval
    - 使用操作符重载来简化基于时间的比较，这样就可以将标准 C++ 语法用于基于时间的算术和关系运算
    - 其构造函数和成员函数将 timeval 结构中的 field 转换成规范的格式，对时间量进行规范化，从而确保在 ACE_Time_Value 对象之间进行精确的比较
    - ACE_Time_Value 可以表示时间长度，比如 5 秒 310000 微秒，也可以绝对的日期和时间，如 2010-12-31 12:09:12

Reactor 框架 - ACE_Time_Value

- ACE_Time_Value 的关键方法
  - 成员函数

方法	描述
ACE_Time_Value() set()	一系列构造函数和 set 方法
sec()	返回 ACE_Time_Value 的秒的部分
usec()	返回 ACE_Time_Value 的微秒的部分
msec()	将 sec()/usec() ACE_Time_Value 格式转换为毫秒格式
operator+=() operator-=() operator*=()	算术方法

友元函数

- operator+ , - , == , != , < , > , <=, >=

Reactor 框架 - ACE_Time_Value

- ACE_Time_Value 类图

```
                    ACE_Time_Value
+ zero : ACE_Time_Value
+ max_time : ACE_Time_Value
- tv_ : timeval
+ ACE_Time_Value (sec : long, usec : long = 0)
+ ACE_Time_Value (t : const struct timeval &)
+ ACE_Time_Value (t : const timespec_t  &)
+ ACE_Time_Value (t : const FILETIME &)
+ set (sec : long, usec : long)
+ set (t : const struct timeval &)
+ set (t : const timespec_t  &)
+ set (t : const FILETIME &)
+ sec () : long
+ usec () : long
+ msec () : long
+ operator+= (tv : const ACE_Time_Value &) : ACE_Time_Value &
+ operator-= (tv : const ACE_Time_Value &) : ACE_Time_Value &
+ operator*= (d : double) : ACE_Time_Value &
```

Reactor 框架 - ACE_Event_Handler

- Reactor 框架：
  - 概要
  - ACE_Time_Value
  - ACE_Event_Handler
  - ACE_Timer_Queue
  - ACE_Reactor
  - Reactor 框架示例
  - Reactor 实现

Reactor 框架 - ACE_Event_Handler

- 关于 ACE_Event_Handler
  - ACE_Event_Handler 是 ACE 中的所有反应式事件处理器的基类，该类的能力：
    - 定义输入、输出事件、异常事件、定时器时间、信号事件的挂钩方法
    - 其挂钩方法允许应用以多种方式扩展事件处理器子类，且不改变框架
    - 由于面向对象回调的使用简化了数据与处理数据的操作关联
    - 对对象的使用还使“事件源”与“事件源关联的数据（如网络会话）”的绑定得以自动化
    - 可以将不再需要的事件处理器的销毁集中在一起
    - 它持有一个指向管理它的 Reactor 的指针，从而使事件处理器对其时间登记和解除登记的正确管理变得很简单

Reactor 框架 - ACE_Event_Handler

- ACE_Event_Handler 关键方法

方法	描述
ACE_Event_Handler()	指定可与事件处理器相关联的 ACE_Reactor 指针
~ACE_Event_Handler()	调用 purge_pending_notifications() 来将其自身从 Reactor 的通知机制中移除
handle_input()	在输入事件发生时被调用的挂钩方法
handle_output()	在输出事件成为可能时（如在流程控制缓和或非阻塞式连接完成时）被调用的挂钩方法
handle_exception()	在异常事件（如 TCP 紧急数据到达）发生时被调用的挂钩方法
handle_timeout()	定时器到期时被调用的方法
handle_signal()	在 OS 发出信号时被调用的挂钩方法
handle_close()	在其它的 handle_* 函数中的一个返回 -1 时，或是在 ACE_Reactor::remove_handler() 被显式调用来解除事件处理器的登记时，执行用户定义的终止操作的挂钩方法
get_handle()	返回底层的 I/O 句柄
reactor()	返回 ACE_Reactor 指针
priority()	获取或设置事件处理器的优先级

Reactor 框架 - ACE_Event_Handler

- 定义 ACE_Event_Handler 挂钩方法需注意：
  - 事件类型和事件处理器挂钩方法
    当应用向 Reactor 登记事件处理器时，它必须指定事件处理器要处理的一个或多个事件的类型。以下是事件类型的枚举：

事件类型	描述
READ_MASK	输入事件，如数据出现在 socket 或文件句柄上。 Reactor 分派 handle_input() 挂钩方法来处理输入事件
WRITE_MASK	输出事件，如流程控制缓和时。 Reactor 分派 handle_output() 挂钩方法来处理输出事件
EXCEPT_MASK	异常事件，如紧急数据 (out of bound) 出现在 socket 上， Reactor 分派 handle_exception() 挂钩方法来处理异常事件
ACCEPT_MASK	被动模式的连接事件 (accept)， Reactor 分派 handle_input() 挂钩方法来处理连接事件
CONNECT_MASK	非阻塞式连接完成， Reactor 分派 handle_output() 挂钩方法来处理非阻塞式连接完成事件

Reactor 框架 - ACE_Event_Handler

- 定义 ACE_Event_Handler 挂钩方法需注意（续 1 ）：
  - 关于事件处理器挂钩方法的返回值
    当所登记的事件发生时， Reactor 分派适当的事件处理器的 handle_*() 挂钩方法来对它们进行处理，关于返回值：
    - 返回值 0 ： 指示 Reactor 应该继续为此事件处理器检测和分派所登记的事件
    - 返回值大于 0 ： 指示 Reactor 应该继续为此事件处理器检测和分派已登记的事件，另外，如果在处理了某个 I/O 事件之后返回的值大于 0 ， Reactor 将在阻塞在它的事件多路分离器上之前，再次在句柄上分派此事件处理器
    - 返回值 -1 ： 指示 Reactor 停止为此事件处理器检测已登记的事件，在 Reactor 从其内部移除此事件处理器之前，先调用处理器的 handle_close()

Reactor 框架 - ACE_Event_Handler

- 定义 ACE_Event_Handler 挂钩方法需注意（续 2 ）：
  - 清理事件处理器
    事件处理器的 handle_close() 方法在其它的某个挂钩方法决定要进行清理时被调用。 handle_close() 方法可以随时进行用户定义的关闭活动

除上表列举的 5 中 MASK， Reactor 还可能将下列枚举值传给 handle_close()：

事件类型	描述
TIMER_MASK	时间驱动事件，由 Reactor 在 handle_timeout() 挂钩方法返回 -1 时传入
WRITE_MASK	信号驱动时间，由 Reactor 在 handle_signal() 挂钩方法返回 -1 时传入

Reactor 框架 - ACE_Event_Handler

- ACE_Event_Handler 类图

```
                    ACE_Event_Handler
- priority_ : int
- reactor_ : ACE_Reactor *
# ACE_Event_Handler (r : ACE_Reactor * = 0,
                     prio : int = LO_PRIORITY)
+ ~ACE_Event_Handler ()
+ handle_input (h : ACE_HANDLE = ACE_INVALID_HANDLE) : int
+ handle_output (h : ACE_HANDLE = ACE_INVALID_HANDLE) : int
+ handle_exception (h : ACE_HANDLE = ACE_INVALID_HANDLE) : int
+ handle_timeout (now: ACE_Time_Value &, act : void * = 0) : int
+ handle_signal (signum : int, info : siginfo_t * = 0,
                 ctx : ucontext_t * = 0) : int
+ handle_close (h : ACE_HANDLE, mask : ACE_Reactor_Mask) : int
+ get_handle () : ACE_HANDLE
+ reactor () : ACE_Reactor *
+ reactor (r : ACE_Reactor *)
+ priority () : int
+ priority (prio : int)
```

Reactor 框架 - ACE_Timer_Queue

- Reactor 框架：
  - 概要
  - ACE_Time_Value
  - ACE_Event_Handler
  - ACE_Timer_Queue
  - ACE_Reactor
  - Reactor 框架示例
  - Reactor 实现

Reactor 框架 - ACE_Timer_Queue

- 关于 ACE_Timer_Queue
  - ACE_Timer_Queue 允许应用登记派生自 ACE_Event_Handler 的时间驱动的时间处理器，这些类提供以下能力：
    - 允许应用调度事件处理器，其 handle_timeout() 方法将在调用者指定的时间被高效而可伸缩的分派：一次性地、或周期性地
    - 允许应用取消与特定事件处理器相关联的某个定时器，或是与某个事件处理器相关联的所有定时器
    - 允许应用配置定时器队列的时间源，如 ACE_OS::gettimeofday() 或 ACE_High_Res_Timer::gettimeofday_hr()

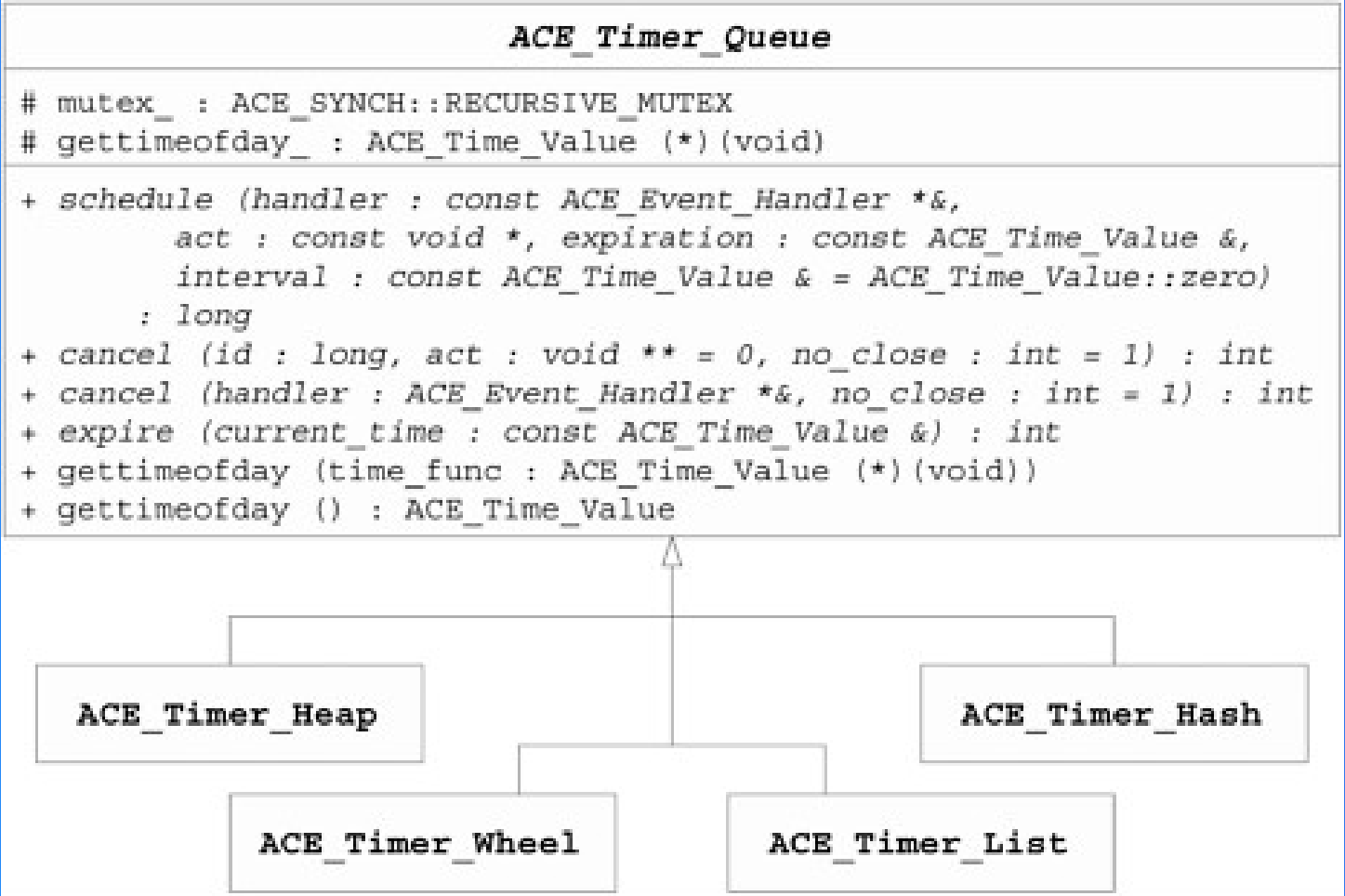
Reactor 框架 - ACE_Timer_Queue

- ACE_Timer_Queue 关键方法

方法	描述
schedule()	调度事件处理器，其 handle_timeout() 方法将在调用者指定的时间被分派，可以选择周期性的进行分派
cancel()	取消与特定事件处理器相关联的定时器，或是与某个事件处理器相关联的所有定时器
expire()	分派到期时间早于或等于当前时间的所有事件处理器的 handle_timeout() 方法，当前时间表示为绝对值，如：2010-12-31 12:08:12
gettimeofday()	两个重载函数，设置 / 获取当前的绝对时间

Reactor 框架 - ACE_Timer_Queue

- ACE_Timer_Queue 类图



Reactor 框架 - ACE_Reactor

- Reactor 框架：
  - 概要
  - ACE_Time_Value
  - ACE_Event_Handler
  - ACE_Timer_Queue
  - ACE_Reactor
  - Reactor 框架示例
  - Reactor 实现

Reactor 框架 - ACE_Reactor

- 关于 ACE_Reactor
  - ACE_Reactor 类是 Reactor 框架的核心，具备以下能力：
    - 使反应式应用中的事件循环处理集中在一起
    - 通过事件多路分离器（如 select() 或 WaitForMultipleObjects()）来检测事件，这些多路分离器由操作系统提供， Reactor 的实现使用它们
    - 在事件多路分离器指示有指定的事件发生时，将事件多路分离给事件处理器
    - 分派已登记的事件处理器上的适当的挂钩方法来执行应用所定义的处理，对事件做出响应
    - 确保任何线程都可以改变反应器的事件集，或是排队对事件处理器的回调，且期望反应器会迅速地对请求做出处理

Reactor 框架 - ACE_Reactor

- ACE_Reactor 类图

```
                          ACE_Reactor
# reactor_ : ACE_Reactor *
# implementation_ : ACE_Reactor_Impl *
+ ACE_Reactor (implementation : ACE_Reactor_Impl * = 0,
               delete_implementation : int = 0)
+ open (max_handles : int, restart : int = 0,
        sig_handler : ACE_Sig_Handler * = 0,
        timer_queue : ACE_Timer_Queue * = 0) : int
+ close () : int
+ register_handler (handler : ACE_Event_Handler *,
                    mask : ACE_Reactor_Mask) : int
+ register_handler (io: ACE_HANDLE, handler: ACE_Event_Handler *,
                    mask : ACE_Reactor_Mask) : int
+ remove_handler (handler : ACE_Event_Handler *,
                  mask : ACE_Reactor_Mask) : int
+ remove_handler (io : ACE_HANDLE, mask : ACE_Reactor_Mask) : int
+ remove_handler (hs : const ACE_Handle_Set&, m :
ACE_Reactor_Mask) : int
+ suspend_handler (handler : ACE_Event_Handler *) : int
+ resume_handler (handler : ACE_Event_Handler *) : int
```

反应器的构造与析构

事件处理器管理

Reactor 框架 - ACE_Reactor

- ACE_Reactor 类图（续 1 ）

```
+ mask_ops (handler : ACE_Event_handler *,
             mask : ACE_Reactor_Mask, ops : int) : int
+ schedule_wakeup (handler : ACE_Event_Handler *,
             masks_to_be_added : ACE_Reactor_Mask ) : int
+ cancel_wakeup (handler : ACE_Event_Handler *,
             masks_to_be_cleared : ACE_Reactor_Mask) : int
+ handle_events (max_wait_time : ACE_Time_Value * = 0) : int
+ run_reactor_event_loop (event_hook : int (*)(void *) = 0) : int
+ end_reactor_event_loop () : int
+ reactor_event_loop_done () : int
+ schedule_timer (handler : ACE_Event_Handler *, arg : void *,
          delay : ACE_Time_Value &,
          repeat : ACE_Time_Value & = ACE_Time_Value::zero) : int
+ cancel_timer (handler : ACE_Event_Handler *,
                dont_call_handle_close : int = 1) : int
+ cancel_timer (timer_id : long, arg : void ** = 0,
                dont_call_handle_close : int = 1) : int
```

事件循环管理

定时器管理

Reactor 框架 - ACE_Reactor

- ACE_Reactor 类图（续 2 ）

```
+ notify (handler : ACE_Event_Handler * = 0,
          mask : ACE_Reactor_Mask = ACE_Event_Handler::EXCEPT_MASK,
          timeout : ACE_Time_Value * = 0) : int
+ max_notify_iterations (iterations : int) : int
+ purge_pending_notifications (handler : ACE_Event_Handler *,
                mask : ACE_Reactor_Mask = ALL_EVENTS_MASK) : int
+ instance () : ACE_Reactor *
+ owner (new_owner : ACE_thread_t,
          old_owner : ACE_thread_t * = 0) : int
```

通知

实用方法

Reactor 框架 - ACE_Reactor

- ACE_Reactor 事件处理器管理方法

方法	描述
register_handler()	登记事件处理器
remove_handler()	移除事件处理器
suspend_handler()	暂停分派事件给某个事件处理器
resume_handler()	恢复被暂停的事件处理器
mask_ops()	获取、设置、添加或清除与某事件处理器相关联的事件类型及其分派掩码
scedule_weakup()	将指定的掩码增加到某事件处理器的条目中，该处理器续是已经通过 register_handler() 登记到该反应器
cancel_weakup()	从事件处理器的条目中清除指定的掩码，但并不将此处理器从反应器中移除

Reactor 框架 - ACE_Reactor

- ACE_Reactor 事件循环管理方法

方法	描述
handle_events()	等待事件发生，并随即分派与之相关联的事件处理器，可以通过超时参数限制消耗在等待事件上的时间
run_reactor_event_loop()	反复调用 handle_ecents() 方法，直至其失败、或是 reactor_event_loop_done() 返回 1，或是发生超时
end_reactor_event_loop()	指示反应器关闭其事件循环
reactor_event_loop_done()	在反应器的事件循环被 end_reactor_event_loop() 调用结束时返回 1

- ACE_Reactor 定时器管理方法

方法	描述
schedule_timer()	登记一个事件处理器，它将在用户指定的时间之后被执行
cancel_timer()	取消一个之前登记的定时器

Reactor 框架 - ACE_Reactor

- ACE_Reactor 通知方法

方法	描述
`notify()`	将时间（以及可选的事件处理器）插入反应器的事件检测器中，从而使其在反应器下次等待事件时被处理
`max_notify_iterations()`	设置反应器通知机制中能分派的处理器的最大数目
`purge_pending_notifications()`	从反应器的通知机制中清除指定的事件处理器或所有事件处理器

- ACE_Reactor 实用方法

方法	描述
`instance()`	静态方法，返回指向一个单例（singleton）ACE_Reactor 的指针
`owner()`	使某个线程拥有反应器的事件循环

Reactor 框架 - 示例

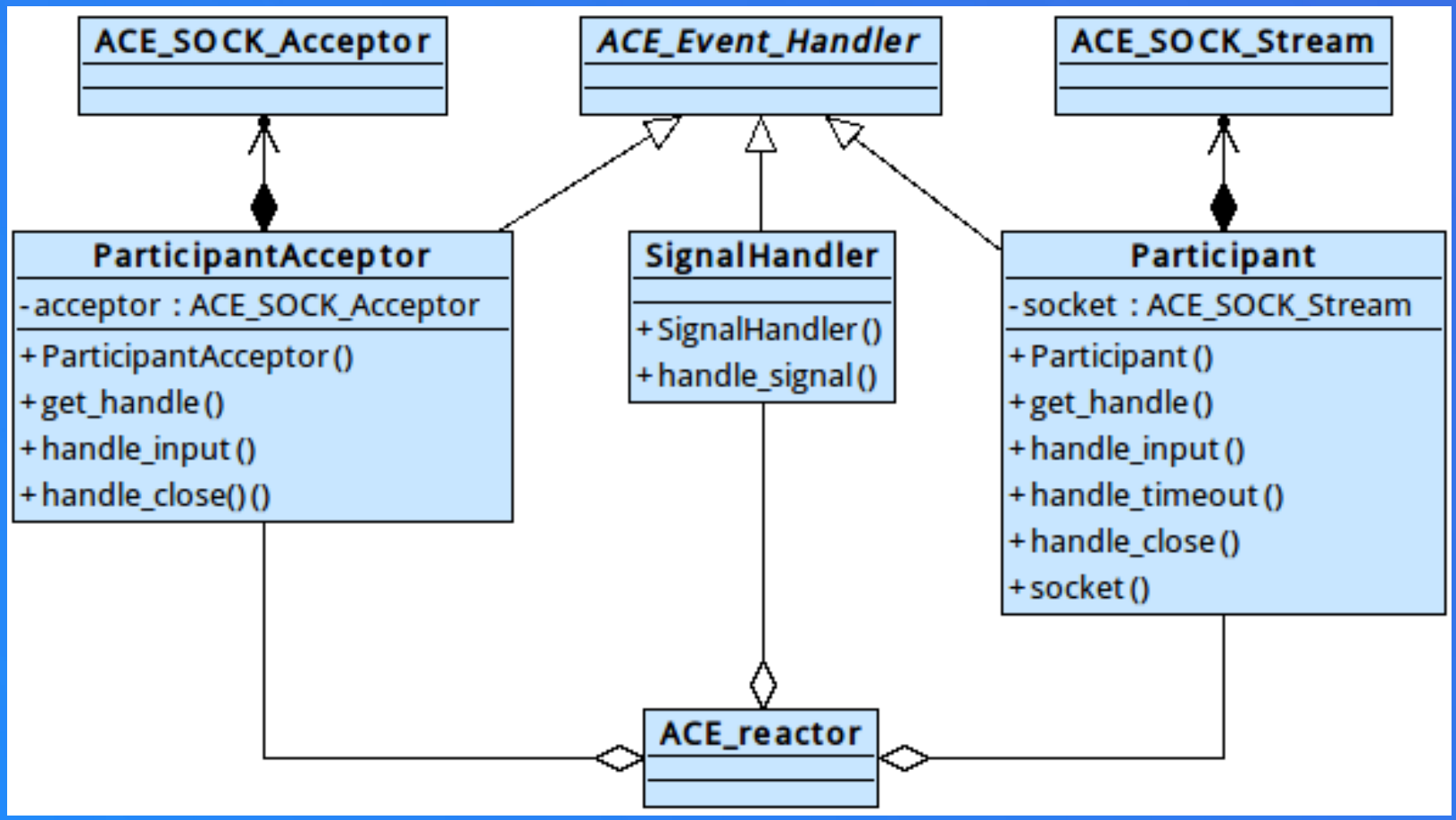
- Reactor 框架：
  - 概要
  - ACE_Time_Value
  - ACE_Event_Handler
  - ACE_Timer_Queue
  - ACE_Reactor
  - Reactor 框架示例
  - Reactor 实现

Reactor 框架 - 示例

- 一个简易的聊天室服务器
  - 该聊天室服务器将实现如下功能：
    - 允许任意客户端加入聊天室
    - 任何加入聊天室的客户端发送到服务器的消息将会被转发至该聊天室的所有成员
    - 管理聊天室成员：当客户端退出聊天室（包括异常终止、长时间未发送消息被服务器逐出等）时，服务器将该成员删除，并通知其它所有成员
    - 如果客户端超过规定的时间（如 60 秒）没有向服务器发送消息，将被服务器逐出聊天室（关闭连接）

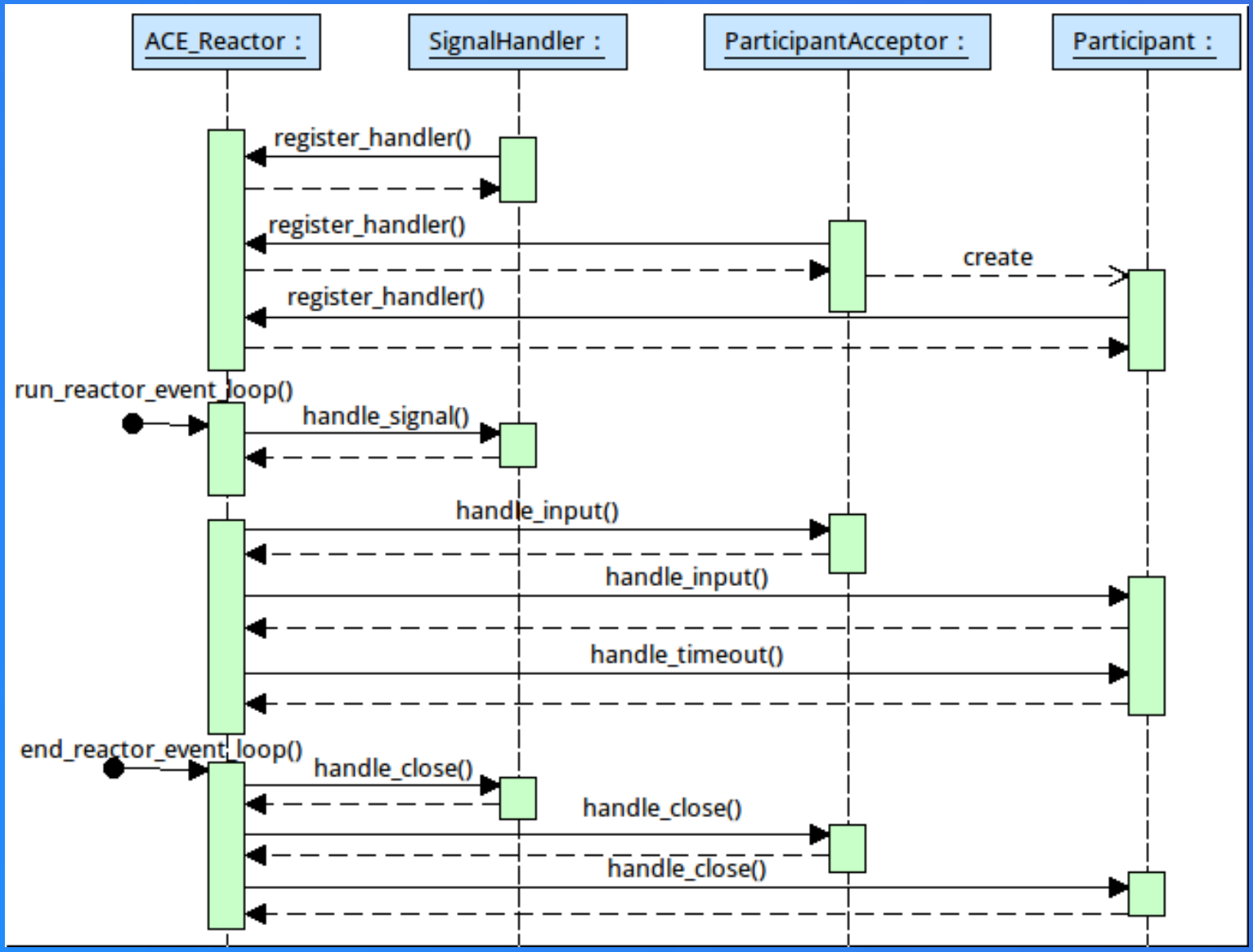
Reactor 框架 - 示例

- 聊天室服务器相关类图



Reactor 框架 - 示例

- 聊天室服务器序列图



Reactor 框架 - 示例

- 聊天室服务器代码
  - （见 Demo 代码）

Reactor 框架 - Reactor 实现

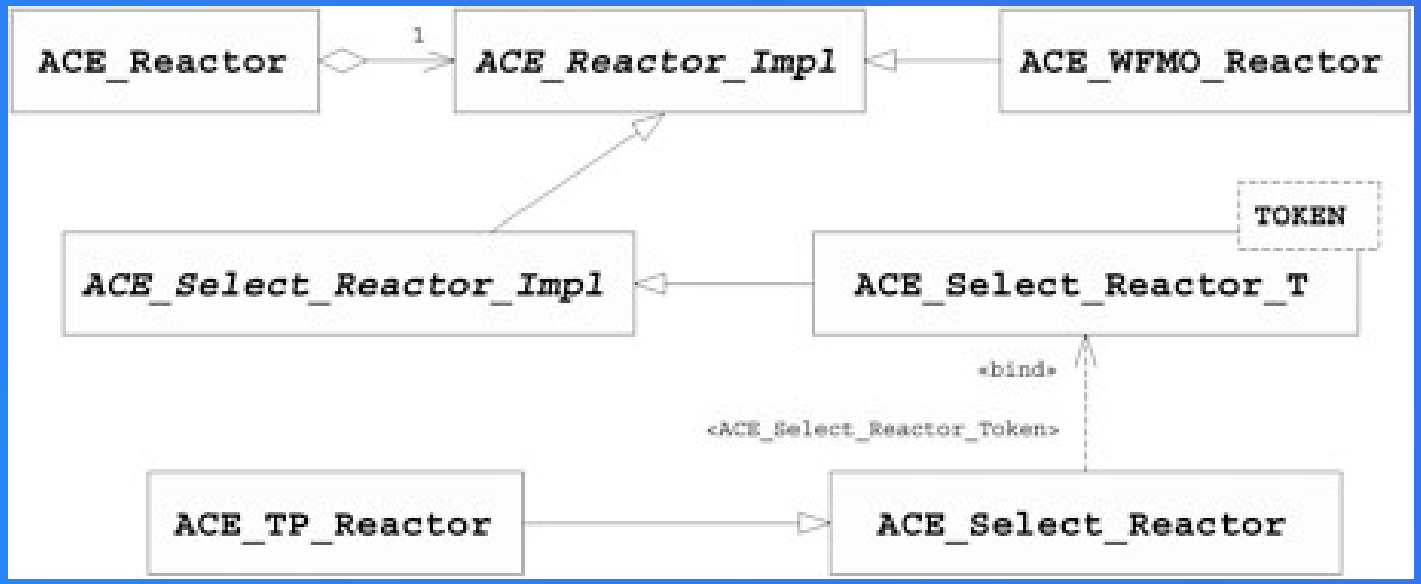
- Reactor 框架：
  - 概要
  - ACE_Time_Value
  - ACE_Event_Handler
  - ACE_Timer_Queue
  - ACE_Reactor
  - Reactor 框架示例
  - Reactor 实现

Reactor 框架 - Reactor 实现

- Reactor 三种常用的实现
  - ACE_Select_Reactor
    - 在非 Windows 平台， Reactor 默认使用 ACE_Select_Reactor，使用 select() 同步事件多路分离函数（或其它高效的分离机制，如 epoll 等）来检测 I/O 、定时器事件，也包含处理 Posix 信号
  - ACE_TP_Reactor
    - 该实现是一个基于 ACE_Select_Reactor 的线程池，采用 Leader/Follows 线程模型
  - ACE_WFMO_Reactor
    - 使用 windows 下的 WaitForMultipleObjects() 事件多路分离函数来检测 socket I/O 、超时事件，以及 windows 同步事件

Reactor 框架 - Reactor 实现

- Reactor 实现图示



Reactor 框架 - Reactor 实现

- 为 reactor 指定不同的实现
  - 将聊天室的 Reactor 实现指定为 ACE_TP_Reactor，提供一个拥有 3 个线程的线程池来检测、分派 I/O 等事件

```
#include <ace/Reactor.h>
#include <ace/TP_Reactor.h>
#include <ace/Thread_Manager.h>
#include <ace/Auto_Ptr.h>

#include "ParticipantAcceptor.h"

// 线程函数
static ACE_THR_FUNC_RETURN eventLoop(void *arg) {
    ACE_Reactor *reactor = static_cast<ACE_Reactor*> (arg);

    reactor->owner(ACE_OS::thr_self());
    reactor->run_reactor_event_loop();
    return 0;
}
```

Reactor 框架 - Reactor 实现

- 为 Reactor 指定不同的实现（续）

```
int main() {
    ParticipantAcceptor acceptor(ACE_Reactor::instance());
    ACE_INET_Addr serverAddr(8868);
    if (acceptor.open(serverAddr) == -1)
        return 1;

    const size_t N_THREADS = 3;
    ACE_Thread_Manager::instance()->spawn_n(N_THREADS,
                    eventLoop, ACE_Reactor::instance());

    ACE_Thread_Manager::instance()->wait();
    return 0;
}
```

Reactor 框架 - 小结

- 在某种意义上讲， Reactor 框架是 ACE 的灵魂
- 这个优秀的框架极大程度上简化了高性能网络应用开发的工作，开发者只需编写极少与业务逻辑无关的代码，将主要的精力集中在业务建模上
- 本次课程通过一个完整的示例项目，全面的展示了 Reactor 框架的应用，包括处理 I/O 事件、定时器事件、信号处理等