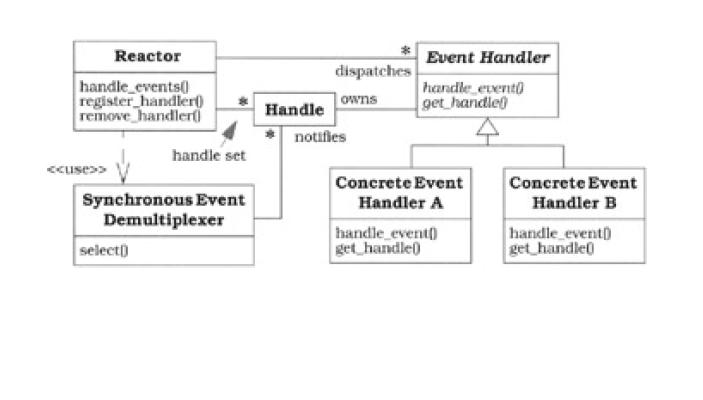
**reactor设计模式**

reactor设计模式，是一种基于事件驱动的设计模式。

 这个模式的结构图如下：



 图中的handle对应的是操作系统提供的句柄，例如I/O句柄，Event\_Handler类持有这些句柄，（其实就是事件，可以看成是Type）

**Handles** ：表示**操作系统管理的资源，我们可以理解为fd**

**Synchronous Event Demultiplexer** ：**同步事件分离器，阻塞等待Handles中的事件发生**

**Initiation Dispatcher** ：初始分派器，作用为添加Event handler（事件处理器）删除Event handler以及分派事件给Event handler也就是说，Synchronous Event Demultiplexer负责等待新事件发生，事件发生时通知Initiation Dispatcher，然后Initiation Dispatcher调用event handler处理事件

**Event Handler** ：事件处理器的接口

**Concrete Event Handler** ：**事件处理器的实际实现，而且绑定了一个Handle**因为在实际情况中，我们往往不止一种事件处理器，因此这里将事件处理器接口和实现分开，与C++Java这些高级语言中的多态类似

过程如下：

 reactor类内部提供一个事件循环：**handle\_events()，事件循环的代码实现利用了操作系统提供的多路分离函数（Select，poll，epoll等），**

 WaitForMultipleObjects或者select等，这些多路分离的函数的特点是，可以同时等待多个句柄，在等待过程中所在线程属于挂起状态（阻塞），不消耗CPU时间，一旦某个句柄被触发，则线程被唤醒，函数将返回，线程可以执行后面的代码，

利用多路分离函数的这一特点，根据被激活的句柄对应的特定事件，调用相关的事件处理函数。可以实现事件循环。

 register\_handler()函数用于将Event\_Handler对象注册到事件驱动列表 中，保证对于某一类型的事件，会调用Event\_Handler类的响应函数handle\_event()。

reactor类在做多路分离时需要操纵Event\_Handler类的Handle，因此Event\_Handler类需要提供get\_handle()函数。

另外，当程序不需要再对特定事件响应时，需要把Event\_Handler对象从事件驱动列表中删除，因此reactor类还实现了 remove\_handler函数。

 因为reactor相对稳定，一旦实现，不需要再定制，所以没有提供一个抽象接口类，但Event\_Handler是经常需要根据不同 的需求定制的，因此需要提供一个抽象接口类，然后根据实际需求编写派生类，提供具体句柄，并实现相关虚函数。

 这个模式的优点是本身不涉及多线程，从而避免了线程的上下文切换。

 对于响应事件处理时间较短的情况下，可以考虑使用这个模式。

 如果处理一个事件需要花费大量时间，就不能使用这个模式，那样会导致其他事件处理被阻塞。

 ACE\_Reactor框架是这一模式的半成品，用户只要做三件事情就可以实现并使用这一模式：

**1.从Event\_Handler派生一个或多个类**

**2.向Reactor类登记应用的事件处理对象**

**3.运行Reactor事件循环（它就会调用IO分离函数，开始轮讯事件，等到有事件返回，传给事件处理者）。**