第1章 ACE自适配通信环境

ACE自适配通信环境 (Adaptive Communication Environment)是面向对象的构架和工具包，它为通信软件实现了核心的并发和分布式模式。ACE包含的多种组件可以帮助通信软件的开发获得更好的灵活性、效率、可靠性和可移植性。ACE中的组件可用于以下几种目的：

- 并发和同步
- 进程间通信(IPC)
- 内存管理
- 定时器
- 信号
- 文件系统管理
- 线程管理
- 事件多路分离和处理器分派
- 连接建立和服务初始化
- 软件的静态和动态配置、重配置
- 分层协议构建和流式构架
- 分布式通信服务：名字、日志、时间同步、事件路由和网络锁定，等等。

1.1 ACE体系结构

如图1-1所示，ACE具有分层的体系结构。在ACE构架中有三个基本层次：

- 操作系统（OS）适配层
- C++包装层
- 构架和模式层

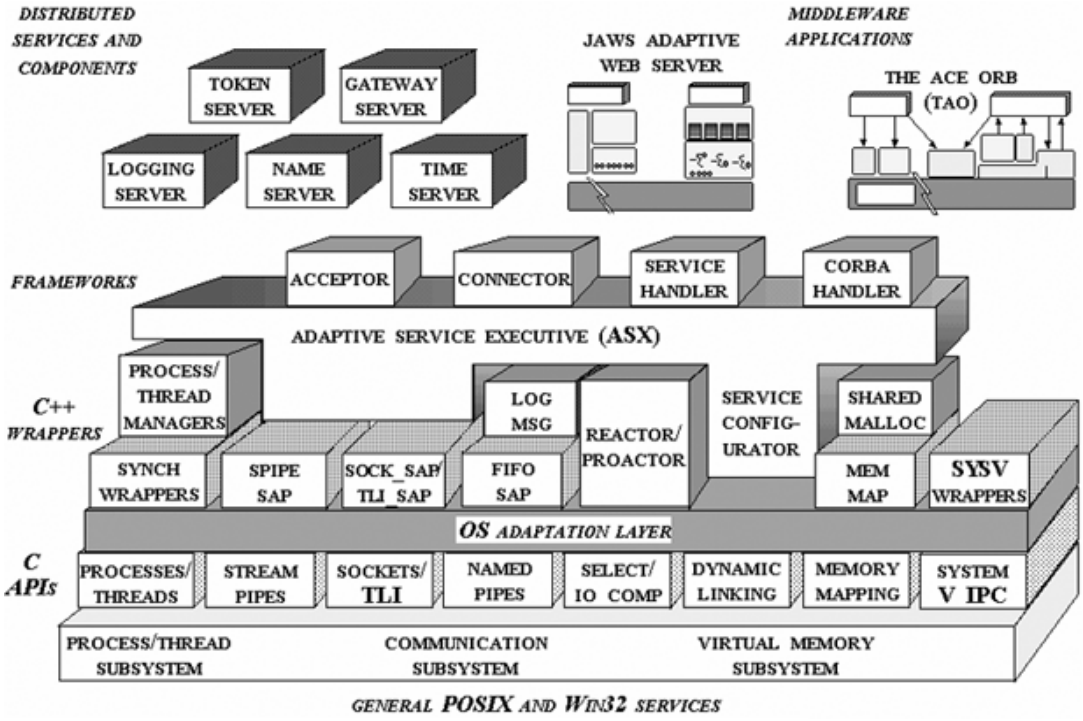

图1-1 ACE的体系结构

1.1.1 OS适配层

OS适配层是位于本地OS API和ACE之间的“瘦”代码层，它使ACE的较高层与平台依赖性屏蔽开来，从而使得通过ACE编写的代码保持了相对的平台无关性。只需要极少的努力，开发者就可以将ACE应用移植到任何平台上。

OS适配层也是ACE构架之所以可用于如此多的平台的原因所在。目前ACE适用的OS平台包括：实时OS（VxWorks、Chorus、LynxOS和pSoS）、大多数版本的UNIX（SunOS 4.x和5.x; SGI IRIX 5.x和6.x; HP-UX 9.x, 10.x和11.x; DEC UNIX 3.x和4.x; AIX 3.x和4.x; DG/UX; Linux; SCO; UnixWare; NetBSD和FreeBSD）、Win32（使用MSVC++和Borland C++的WinNT 3.5.x、4.x、Win95和WinCE）以及MVS OpenEdition。

1.1.2 C++包装层

C++包装层包括一些C++包装类，它们可用于构建高度可移植的和类型安全的C++应用。这是ACE工具包最大的一部分，大约包含了总源码的50%。C++包装类可用于：

- **并发和同步：**ACE提供若干并发和同步包装类，对本地OS多线程和多进程API进行了抽象。这些包装类封装用于线程和进程的原语，比如信号量、锁、栅栏（Barrier）和条件变量。另外还有更高级的原语可用，比如守卫（Guard）。所有这些原语共享类似的接口，因而很容易使用和相互替换。
- **IPC：**ACE提供若干C++包装类，封装不同OS中不同的进程间通信（IPC）接口。例如，ACE的包装类封装了以下IPC机制：BSD socket、TLI、UNIX FIFO、流管道、Win32命名管道，等等。ACE还为消息队列提供包装类，包括特定的实时OS的消息队列。
- **内存管理组件：**ACE包含的一些类可用于内存动态分配和释放；其中包括允许预分配所有动态内存的类。这些预分配的内存随即通过ACE提供的管理类的帮助进行本地管理。在大多数实时和嵌入式系统中，这样的细粒度管理极为必要。另外还有一些类用于灵活地管理进程间共享内存。

- **定时器类：**有多种不同的类可用于处理定时器的调度和取消。ACE中不同种类的定时器使用不同的底层机制（堆、定时器轮（timer wheel）或简单列表）来提供不同的性能特性。但是，不管底层使用何种机制，这些类的接口都是一致的，从而使得开发者很容易使用任何一种定时器类。除了这些定时器类，还有封装高分辨率定时器（在部分平台上可用，比如VxWorks, Win32/Pentium, AIX和Solaris）和Profile Timer的包装类。
- **容器类：**ACE还拥有若干可移植的STL风格的容器类，比如Map、Hash_Map、Set、List，等等
- **信号处理：**ACE提供对特定OS的信号处理接口进行封装的包装类。这些类使得开发者能够很容易地安装和移除信号处理器，并且可以为一个信号安装若干处理器。另外还有信号守卫类，可用于在看守的作用域之内禁止所有信号。
- **文件系统组件：**ACE含有包装文件系统API的类。这些类包括文件I/O、异步文件I/O、文件加锁、文件流、文件连接包装，等等。
- **线程管理：**ACE提供包装类来创建和管理线程。这些包装还封装了针对特定OS的线程API，可被用于提供像线程专有存储这样的功能。

1.1.3 ACE构架组件

ACE构架组件是ACE中最高级的“积木”，它们的基础是若干针对特定通信软件领域的设计模式。设计者可以使用这些构架组件来帮助自己在高得多的层面上思考和构建系统。这些组件实际上为将要构建的系统提供了“袖珍体系结构”，因此这些组件不仅在开发的实现阶段、同时在设计阶段都是有用的。ACE的这一层含有以下一些大型组件：

- **事件处理：**大多数通信软件都含有大量处理各种类型事件（比如，基于I/O、基于定时器、基于信号和基于同步的事件）的代码。软件必须高效地多路分离、分派和处理这些事件。遗憾的是，大多数时间开发者们都在反复地编写这些代码，“重新发明轮子”。这是因为，事件多路分离、分派和处理代码全都紧密地耦合在一起，无法彼此独立地使用。ACE提供了被称为**Reactor**（反应堆）的构架组件来解决这一问题。反应堆提供用于高效地进行事件多路分离和分派的代码，并极大地降低了它们与处理代码之间的耦合，从而改善了可复用性和灵活性。
- **连接或服务初始化组件：**ACE提供**Connector**（连接器）和**Acceptor**（接受器）组件，用于降低连接初始化与连接建立后应用执行的实际服务之间的耦合。在接受大量连接请求的应用服务器中，该组件十分有用。连接首先以应用特有的方式初始化，然后每一连接被相应的处理例程分别处理。这样的去耦合使得开发者能够分别去考虑连接的处理和初始化。因此，如果在随后的阶段开发者发现连接请求数多于或是少于估算，它可以选择使用不同的初始化策略集（ACE提供了若干可供开发者挑选和选择的策略），以获得所要求的性能水平。
- **流组件：**ACE **Stream**组件用于简化那些本质上是分层的（layered）或层次的（hierarchic）软件的开发。用户级协议栈的开发是一个好例子；这样的栈由若干互连的层次组成。这些层次或多或少可以相互独立地进行开发。当“数据”通过时，每一层都处理并改变它，并将它传递给下一层，以作进一步的处理。因为各层是相互独立的，它们很容易被复用或替换。
- **服务配置组件：**通信软件开发者面临的另一个问题是，很多时候，软件服务必须在安装时配置，或必须在运行时重配置。应用中特定服务的实现可能需要进行*改动*，因而应用可能必须用新改动的服务重新配置。ACE **Service Configurator**（服务配置器）为应用的服务提供动态的启动、挂起和配置。

尽管计算机网络领域发展迅速，编写通信软件已经变得更为困难。大量消耗在开发通信软件上的努力不过是“重新发明轮子”的变种，已知的可以在应用间通用的组件被重写，而不是被复用。通过收集通用的组件和体系结构（它们在网络和系统编程领域一再被复用），ACE为这一问题提供了解决方案。

应用开发者可以采用ACE，挑选和选择在他的应用中所需的组件，并开始在ACE工具箱的陪伴下构建应用。除了在C++包装层中收集简单的“积木”，ACE还包括了大的体系结构“积木”，它们采用了已被证明在软件开发领域中行之有效的“模式”和“软件体系结构”。