前言（节选）

本书主要讲述采用现代在上编写多线程网络服务程序的主流常规技术，这也是我对过去 5 年编写生产环境下的多线程服务端程序的经验总结。本书重点讲解多线程网络服务器的一种模型，即。这是一种适应性较强的模型，也是下以语言编写用户态高性能网络程序最成熟的模式，掌握之后可顺利地开发各类常见的服务端网络应用程序。本书以网络库为例，讲解这种编程模型的使用方法及注意事项。

是一个基于非阻塞和事件驱动的现代网络库，原生支这种模型。适合开发下的面向业务的多线程服务端网络应用程序，其中“面向业务的网络编程”的定义见附录。“现代”指的不是新标准，而是发布之后的语言和库。 与传统相比，现代的变化主要有两方面：资源管理（见第 1 章）与事件回调（见第 449 页）。

本书不是多线程编程教程，也不是网络编程教程，更不是教程。读者应该已经大致读过主流的服务端发行版的版本都还停留在进入实用尚需一段时日。

本书适用的硬件环境是主流 x86-64 服务器，多路多核 CPU、几十 GB 内存、千兆以太网互联。除了第 5 章讲诊断日志之外，本书不涉及文件 IO。

本书分为四大部分，第 1 部分“C++ 多线程系统编程”考察多线程下的对象生命期管理、线程同步方法、多线程与 C++ 的结合、高效的多线程日志等。第 2 部分“muduo 网络库”介绍使用现成的非阻塞网络库编写网络应用程序的方法，以及 muduo 的设计与实现。第 3 部分“工程实践经验谈”介绍分布式系统的工程化开发方法和 C++ 在工程实践中的功能特性取舍。第 4 部分“附录”分享网络编程和 C++ 语言的学习经验。

本书的宗旨是贵精不贵多。掌握两种基本的同步原语就可以满足各种多线程同步的功能需求，还能写出更易用的同步设施。掌握一种进程间通信方式和一种多线程网络编程模型就足以应对日常开发任务，编写运行于公司内网环境的分布式服务系统。（本书不涉及分布式存储系统，也不涉及 UDP。）

术语与排版范例

本书大量使用英文术语，甚至有少量英文引文。设计模式的名字一律用英文，例如 Observer、Reactor、Singleton。在中文术语不够突出时，也会使用英文，例如 class、heap、event loop、STL algorithm 等。注意几个中文 C++ 术语：对象实体（instance）、函数重载决议（resolution）、模板具现化（instantiation）、覆写（override）虚函数、提领（dereference）指针。本书中的英语可数名词一般不用复数形式，例如两个 class，6 个 syscall；但有时会用 (s) 强调中文名词是复数。fd 是文件描述符（file

前言（节选）

本书主要讲述采用现代 C++ 在 x86-64 Linux 上编写多线程 TCP 网络服务程序的主流常规技术，这也是我对过去 5 年编写生产环境下的多线程服务端程序的经验总结。本书重点讲解多线程网络服务器的一种 IO 模型，即 one loop per thread。这是一种适应性较强的模型，也是 Linux 下以 native 语言编写用户态高性能网络程序最成熟的模式，掌握之后可顺利地开发各类常

见的服务端网络应用程序。本书以 muduo 网络库为例，讲解这种编程模型的使用方法及注意事项。

muduo 是一个基于非阻塞 IO 和事件驱动的现代 C++ 网络库，原生支持 one loop per thread 这种 IO 模型。muduo 适合开发 Linux 下的面向业务的多线程服务端网络应用程序，其中“面向业务的网络编程”的定义见附录 A。“现代 C++”指的不是 C++11 新标准，而是 2005 年 TR1 发布之后的 C++ 语言和库。与传统 C++ 相比，现代 C++ 的变化主要有两方面：资源管理（见第 1 章）与事件回调（见第 449 页）。

本书不是多线程编程教程，也不是网络编程教程，更不是 C++ 教程。读者应该已经大致读过《UNIX 环境高级编程》、《UNIX 网络编程》、《C++ Primer》或与之内容相近的书籍。本书不谈 C++11，因为目前（2012 年）主流的 Linux 服务端发行版的 g++ 版本都还停留在 4.4，C++11 进入实用尚需一段时日。

本书适用的硬件环境是主流 x86-64 服务器，多路多核 CPU、几十 GB 内存、千兆以太网互联。除了第 5 章讲诊断日志之外，本书不涉及文件 IO。

本书分为四大部分，第 1 部分“C++ 多线程系统编程”考察多线程下的对象生命期管理、线程同步方法、多线程与 C++ 的结合、高效的多线程日志等。第 2 部分“muduo 网络库”介绍使用现成的非阻塞网络库编写网络应用程序的方法，以及 muduo 的设计与实现。第 3 部分“工程实践经验谈”介绍分布式系统的工程化开发方法和 C++ 在工程实践中的功能特性取舍。第 4 部分“附录”分享网络编程和 C++ 语言的学习经验。

本书的宗旨是贵精不贵多。掌握两种基本的同步原语就可以满足各种多线程同步的功能需求，还能写出更易用的同步设施。掌握一种进程间通信方式和一种多线程网络编程模型就足以应对日常开发任务，编写运行于公司内网环境的分布式服务系统。（本书不涉及分布式存储系统，也不涉及 UDP。）

术语与排版范例

本书大量使用英文术语，甚至有少量英文引文。设计模式的名字一律用英文，例如 Observer、Reactor、Singleton。在中文术语不够突出时，也会使用英文，例

如 class、heap、event loop、STL algorithm 等。注意几个中文 C++ 术语：对象实体（instance）、函数重载决议（resolution）、模板具现化（instantiation）、覆写（override）虚函数、提领（dereference）指针。本书中的英语可数名词一般不用复数形式，例如两个 class，6 个 syscall；但有时会用 (s) 强调中文名词是复数。fd 是文件描述符（file

前言（节选）

本书主要讲述采用现代 C++ 在 x86-64 Linux 上编写多线程 TCP 网络服务程序的主流常规技术，这也是我对过去 5 年编写生产环境下的多线程服务端程序的经验总结。本书重点讲解多线程网络服务器的一种 IO 模型，即 one loop per thread。这是一种适应性较强的模型，也是 Linux 下以 native 语言编写用户态高性能网络程序最成熟的模式，掌握之后可顺利地开发各类常见的服务端网络应用程序。本书以 muduo 网络库为例，讲解这种编程模型的使用方法及注意事项。

muduo 是一个基于非阻塞 IO 和事件驱动的现代 C++ 网络库，原生支持 one loop per thread 这种

IO 模型。muduo 适合开发 Linux 下的面向业务的多线程服务端网络应用程序，其中“面向业务的网络编程”的定义见附录 A。“现代 C++”指的不是 C++11 新标准，而是 2005 年 TR1 发布之后的 C++ 语言和库。与传统 C++ 相比，现代 C++ 的变化主要有两方面：资源管理（见第 1 章）与事件回调（见第 449 页）。

本书不是多线程编程教程，也不是网络编程教程，更不是 C++ 教程。读者应该已经大致读过《UNIX 环境高级编程》、《UNIX 网络编程》、《C++ Primer》或与之内容相近的书籍。本书不谈 C++11，因为目前（2012 年）主流的 Linux 服务端发行版的 g++ 版本都还停留在 4.4，C++11 进入实用尚需一段时日。

本书适用的硬件环境是主流 x86-64 服务器，多路多核 CPU、几十 GB 内存、千兆以太网互联。除了第 5 章讲诊断日志之外，本书不涉及文件 IO。

本书分为四大部分，第 1 部分“C++ 多线程系统编程”考察多线程下的对象生命期管理、线程同步方法、多线程与 C++ 的结合、高效的多线程日志等。第 2 部分“muduo 网络库”介绍使用现成的非阻塞网络库编写网

络应用程序的方法，以及 muduo 的设计与实现。第 3 部分“工程实践经验谈”介绍分布式系统的工程化开发方法和 C++ 在工程实践中的功能特性取舍。第 4 部分“附录”分享网络编程和 C++ 语言的学习经验。

本书的宗旨是贵精不贵多。掌握两种基本的同步原语就可以满足各种多线程同步的功能需求，还能写出更易用的同步设施。掌握一种进程间通信方式和一种多线程网络编程模型就足以应对日常开发任务，编写运行于公司内网环境的分布式服务系统。（本书不涉及分布式存储系统，也不涉及 UDP。）

术语与排版范例

本书大量使用英文术语，甚至有少量英文引文。设计模式的名字一律用英文，例如 Observer、Reactor、Singleton。在中文术语不够突出时，也会使用英文，例如 class、heap、event loop、STL algorithm 等。注意几个中文 C++ 术语：对象实体（instance）、函数重载决议（resolution）、模板具现化（instantiation）、覆写（override）虚函数、提领

（dereference）指针。本书中的英语可数名词一般不用复数形式，例如两个 class，6 个 syscall；但有时会用 (s) 强调中文名词是复数。fd 是文件描述符（file