服务器核心架构设计项目

**项目名称：\_服务器核心架构设计（C++实现）**

**项目编号： 00001**

**编写人员： （12组）郭乐晨\_\_\_\_**

**编写日期： 2019.6.27 \_**

[1. 引言 3](#_Toc303775593)

[1.1 编写目的 3](#_Toc303775594)

[1.2 预期读者和阅读建议 3](#_Toc303775596)

[1.3 参考资料 3](#_Toc303775597)

[2. 设计概述 4](#_Toc303775598)

[2.1 限制和约束 4](#_Toc303775599)

[2.2 设计原则和设计要求 4](#_Toc303775600)

[3. 服务器架构概要设计 4](#_Toc303775601)

[3.1 总体架构设计 4](#_Toc303775602)

[3.2 \_include设计 5](#_Toc303775603)

[3.3 app设计 6](#_Toc303775606)

[3.4 net设计 7](#_Toc303775609)

[3.5 proc设计 8](#_Toc303775609)

[3.6 signal设计 8](#_Toc303775609)

[3.7 misc设计 9](#_Toc303775609)

[4. 技术设计 9](#_Toc303775613)

[4.1 系统开发技术说明表 9](#_Toc303775614)

# 1、引言

 本概要设计文档主要依据需求规格说明书，来指导服务器架构的详细设计工作,为详细设计提供统一的参照标准。文档首先给出了整个服务器的整体功能结构的概貌，试图从总体架构上给出服务器的轮廓，然后又对实现各个功能提供详细的概要设计。本文档能详细准确地指导接下来的详细设计工作。

## 编写目的

这份软件系统概要设计报告是基于服务器核心架构设计项目需求规格说明书编写的，开发这个软件产品意义、作用、以及最终要达到的意图。旨在实现一个多线程高并发企业级服务器的核心框架，日后做任何软件服务器都可以基于此框架进行拓展。

## 预期读者和阅读建议

本文档的主要内容共分3部分：设计概述、服务器架构概要设计、技术设计。设计概述部分主要对服务器的整体结构进行了大致的介绍；服务器架构设计对系统的功能需求进行了详细描述，是本文的主要部分；技术设计部分对服务器架构所需采用技术进行了详细的描述。

本文档面向多种读者对象：

（1）设计员：对服务器概要设计文档进行分析，并设计出服务器。

（2）程序员：配合《服务器需求规格说明书》，了解服务器功能。

（3）用户：了解预期产品的功能，并与分析人员一起对整个需求进行讨论和协商。

（4）其他人员：如部门领导、公司领导等可以据此了解产品的功能和性能。

在阅读本文档时，首先要了解产品的功能概貌，然后可以根据自身的需要对每一功能进行适当的了解。

## 参考资料

《服务器需求规格说明书》

《TCP/IP网络编程》

《C++并发编程》

# 设计概述

本节描述现有开发条件和需要实现的目标，说明进行概要设计时应该遵循的设计原则和必须采用的设计方法。

## 限制和约束

技术条件：使用C++开发

资金状况：没有资金支持

开发环境：Linux环境下运行，使用Visual Studio 2017 开发

时间限制：两周

## 设计原则和设计要求

命名规则：代码使用驼峰命名法

实用性：实用性是服务器的主要设计原则，服务器设计必须最大可能地满足用户的需求。

先进性：信息技术发展迅速，服务器设计尽可能采用先进的技术标准和技术方 法。

灵活和易维护：采用开放的体系架构，具备完整的文档说明。

维护方面：对于开发人员来讲，系统编码容易调整。

# 服务器架构概要设计

## 总体架构设计

使用makefile编译项目生成可执行文件

目录结构描述：

\_include目录：专门存放各种头文件。

app目录：放主应用程序以及一些比较核心的文件。

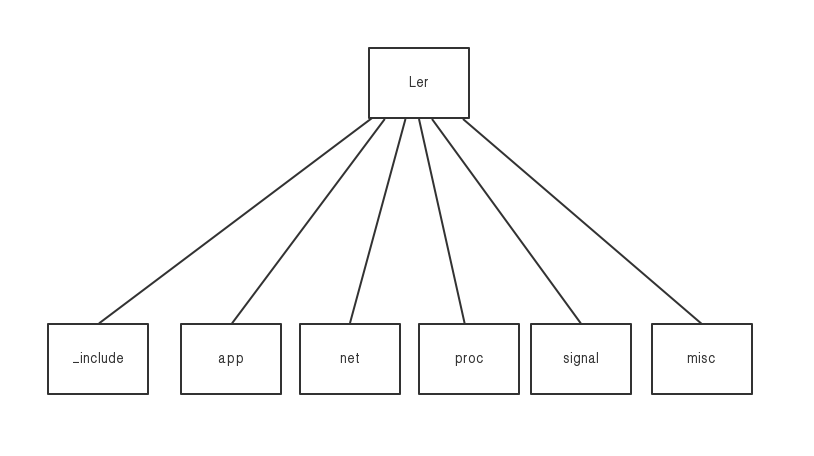
net目录：专门存放和网络处理相关的文件。

proc目录：专门存放和进程处理有关的文件。

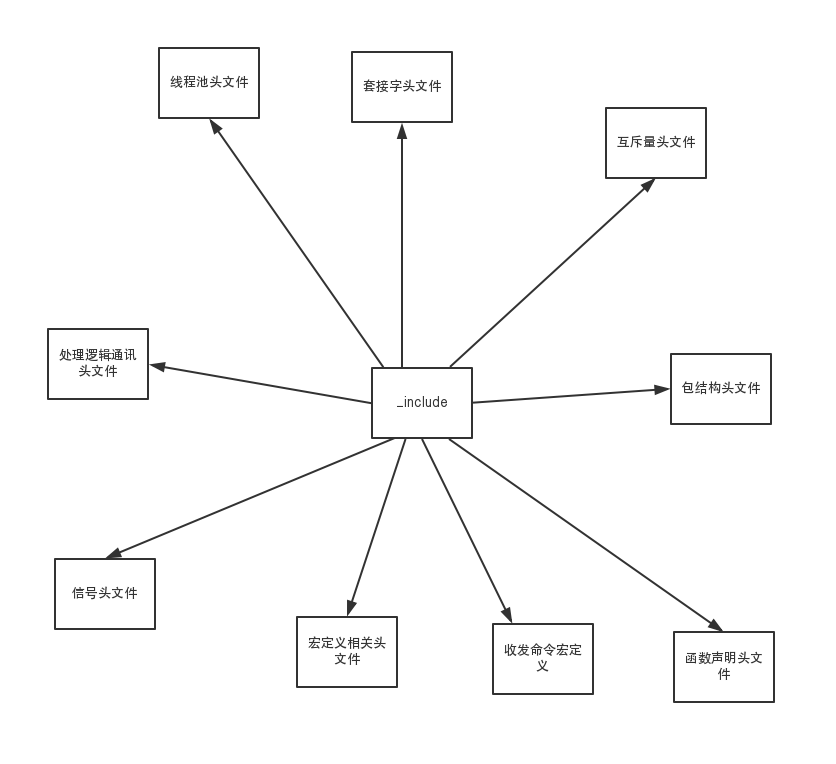
signal目录：专门用于存放和信号处理有关的文件。

misc目录：专门存放各种杂合性的不好归类的文件。

目录结构如下图所示：



## \_include目录中架构设计



## app目录中架构设计

1)link\_obj：临时目录：会存放临时的.o文件，这个目录不手工创建，后续用makefile脚本来创建。

2)Ler\_setproctitle.cxx:设置课执行程序标题（名称）相关.

3)dep：临时目录，会存放临时的.d开头的依赖文件，依赖文件能够告知系统哪些相关的文件发生变化，需要重新编译，后续用makefile脚本来创建。

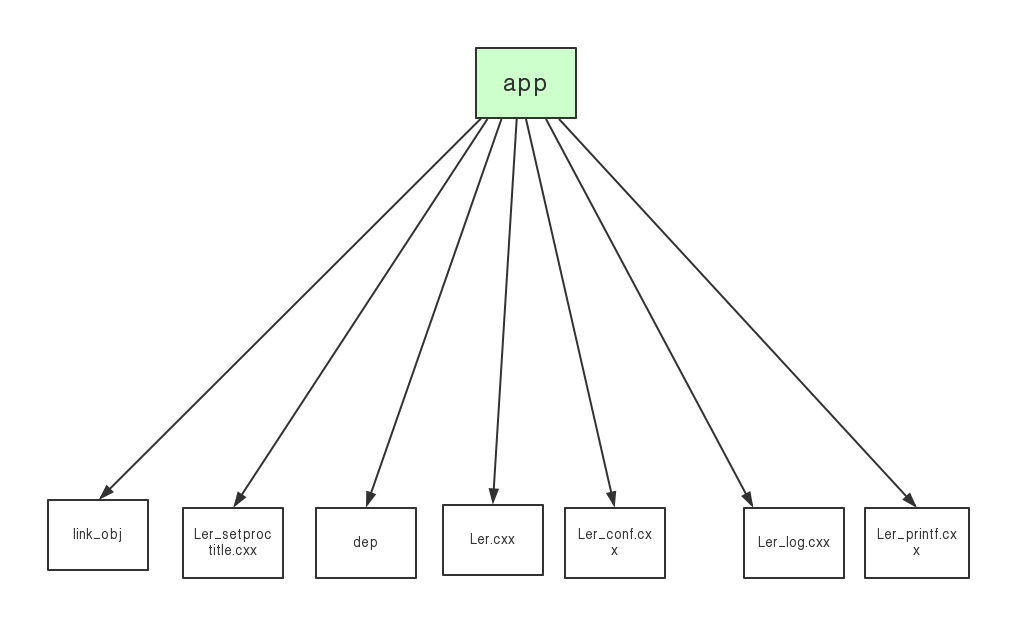
4)Ler.cxx：主文件，main()入口函数就放到这里。

5)Ler\_conf.c：普通的源码文件，跟主文件关系密切，又不值得单独放在 一个目录；

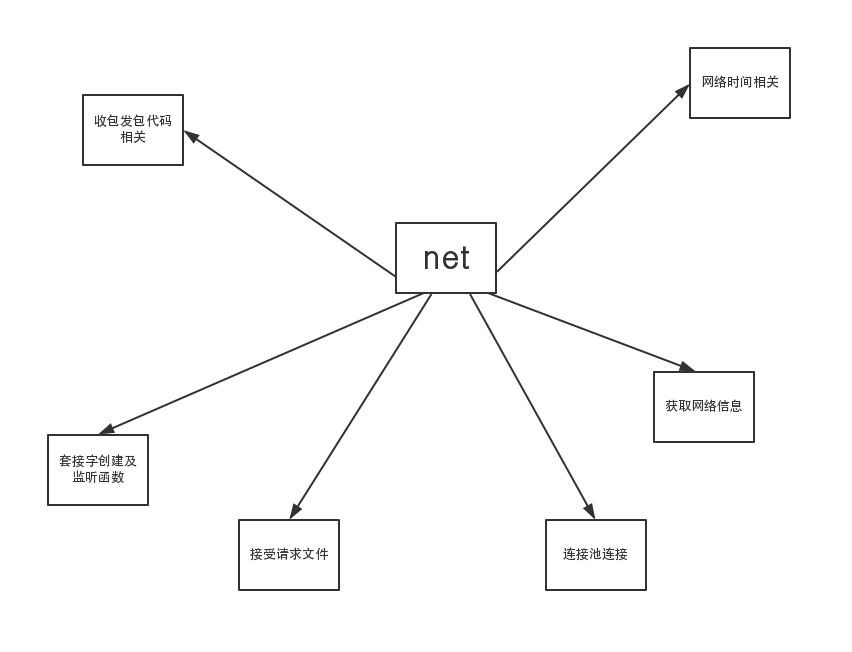
6)Ler\_log.cxx:存放日志相关方法

7）Ler\_printf.cxx:打印格式相关方法

结构如图所示：



## net目录中架构设计



如上图所示，创建文件名依据上到下，从左往右顺序依次为：

Ler\_sokect\_time.cxx

Ler\_sokect\_request.cxx

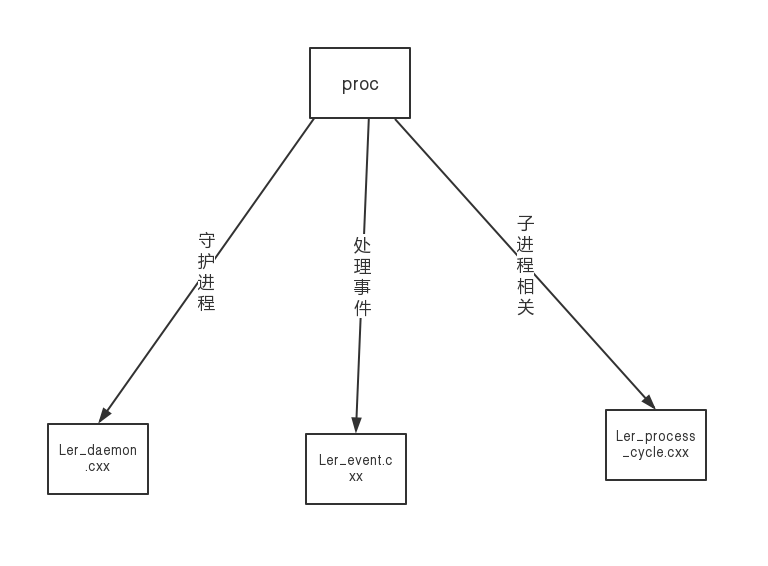
Ler\_sokect\_inet.cxx

Ler\_sokect\_socket.cxx

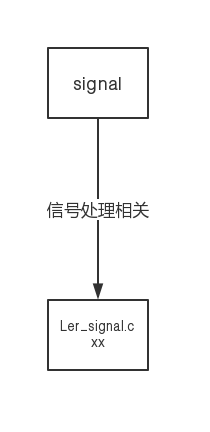
Ler\_sokect\_accept.cxx

Ler\_sokect\_conn.cxx

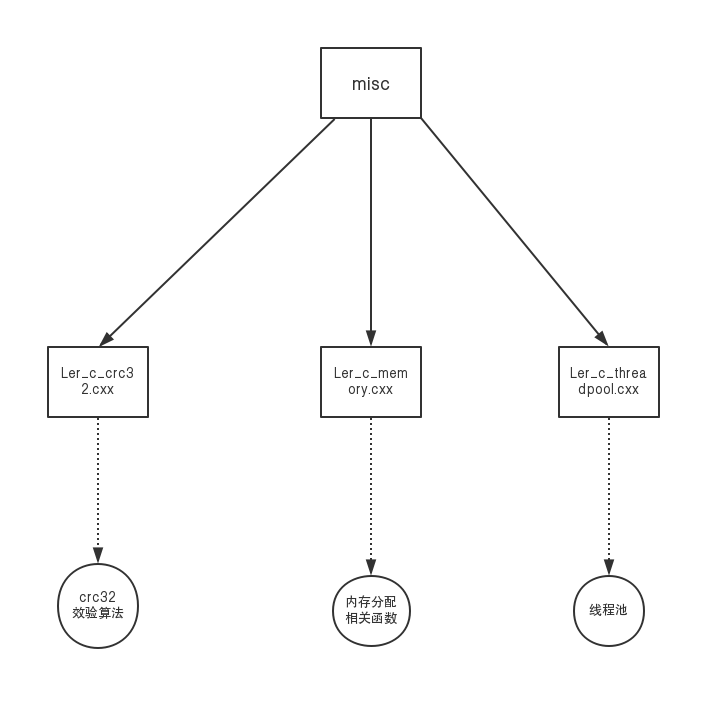
## proc目录中架构设计



## signal目录中架构设计



## misc目录中架构设计



# 技术设计

服务器技术设计描述系统各个特性实际使用的开发技术，以及具体开发技术使用时应该注意的事项。

## 系统开发技术说明表

本表描述系统各个特性开发时实际使用的具体技术，只有一些不太常用的技术需要在这里描述。

系统开发技术说明表如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 技术编号 | 开发技术 | 开发技术 | 处理功能 |
| 英文名称 | 中文名称 |
| 001 | Threadpool | 线程池 | 一种多线程处理形式，处理过程中将任务添加到队列，然后在创建线程后自动启动这些任务。 |
| 002 | Connectionpool | 连接池 | 创建和管理一个连接的缓冲池的技术，这些连接准备好被任何需要它们的线程使用。 |
| 003 | Thread  Synchronization | 线程同步 | 当有一个线程在对内存进行操作时，其他线程都不可以对这个内存地址进行操作，直到该线程完成操作， 其他线程才能对该内存地址进行操作。 |
| 004 | Epoll | 高并发通讯 | 显著提高程序在大量并发连接中只有少量活跃的情况下的系统CPU利用率。 |
| 005 | Makefile | 项目管理 | 定义了一系列的规则来指定，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，还可以进行更复杂的功能操作。 |
| 006 | Singleton | 单例模式 | 一个类只允许拥有一个实例，避免一个全局使用的类频繁地创建与销毁。 |
| 007 | Daemon | 守护进程 | 一类在后台运行的特殊进程，用于执行特定的系统任务。很多守护进程在系统引导的时候启动，并且一直运行直到系统关闭。 |
| 008 | Socket | 套接字编程 | 建立网络通讯连接，双向的通信连接实现数据的交换。 |