学号_____

密级_____

武汉大学本科毕业论文

基于 LAMP 架构的 ACM/ICPC 在线评测系统设计

院（系）名 称：计算机学院

专 业 名 称 ：信息安全

学 生 姓 名 ：冯敏

指 导 教 师 ：董文永 副教授

二〇一〇年六月

BACHELOR'S DEGREE THESIS
OF WUHAN UNIVERSITY

Designation of ACM/ICPC Online Judge System Based on LAMP Architechture

College : Computer School

Subject : Information Security

Name : Feng Min

Directed by : Dong Wenyong Professor

June 2010

郑 重 声 明

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

本人签名：_____　　日期：_____

摘　　要

在线评测系统（Online Judge System，一般简称 OJ），是一个为 ACM/ICPC（国际大学生程序设计竞赛）训练队伍和程序设计竞赛爱好者提供训练及交流的平台，同时也可以为程序设计语言、数据结构、算法等课程提供作业练习、实现及检测的平台。

本课题在武汉大学已有的在线评测系统上进行全面重构，将 OJ 系统各个组成部分进行解耦，使得 OJ 系统各个模块之间独立性增强，容易修改现有功能及扩充新功能，同时也方便组成服务器集群，以应对举办比赛时的访问压力。

本文所讨论的内容包括是本课题的系统架构设计，以及对其下各个模块核心内容的阐述，包括基于 Linux 系统的沙箱模型、基于 Java 的多线程服务器、基于 AMP 架构和 MVC 设计模式的 Web 前端系统的设计与实现，以及影响系统安全的各种因素和对应的解决方案。

本课题的系统实现已经完成，并在武汉大学 ACM/ICPC 集训队的服务器上稳定运行，为本课题提供了充分的实践支持，验证了本文的理论基础和研究成果。

关键词：在线评测系统；沙箱模型；系统安全；ACM/ICPC；LAMP 架构

ABSTRACT

Online Judge System (OJ) is a training platform for ACM/ICPC (International Collegiate Programming Contest) contestants and programming lovers, and it can be used as a homework test platform for the courses like Programming Language, Data Structures, Algorithms which need students to write programs to solve their homework.

This subject reconstructed Oak (the previous version of Online Judge System of Wuhan University), and decoupled its components, which not only increases the independence between the components, but also makes it much easier to modify and expand the system's functionality. What's more, it becomes possible to run clusters which helps relief the heavy pressure brought to the server when a contest in running.

This article discussed the architecture of this OJ system, as well as the core of its components, including the designation and implementation of Linux-based sandbox module, Java-based multi-thread server, Web front-end system based on AMP architecture and MVC design pattern, as well as the various factors affecting the system's security and the corresponding solutions.

The system has already been implemented, and is running stably on the server of ACM/ICPC association of Wuhan University for a long time, which provide adequate practical support to verify the theoretical basis and research results.

Key words: Online Judge System; Sandbox model; System Security; ACM/ICPC; LAMP Architecture

目　录

第 1 章　绪论

1.1　课题背景

ACM/ICPC 从 1970 年开始，已经举办了 30 多届，其目的旨在使大学生运用计算机来充分展示自己分析问题和解决问题的能力。该竞赛一直受到国际各知名大学的重视，并受到全世界各著名计算机公司的高度关注。

武汉大学自 2002 年组建 ACM/ICPC 集训队，至今已经获得了非常优异的成绩，包括三次参与 World Final 世界总决赛，以及数枚亚洲赛区区域赛金牌，数十枚银牌、铜牌等，同时也对武汉大学计算机学院的学风建设起了良好的推动作用。

在 ACM/ICPC 集训队队员的日常训练中，在线评测系统(Online Judge System，以下简称 OJ）起到了非常重要的作用。集训队员可以在线评测系统上挑选各种题目挑战自我，提高自我，学习各种数据结构和算法；在统一组织的集中训练中可以通过指定题目的形式来强化训练效果；而在线评测系统对比赛功能的支持，进一步提高了集训队员的学习热情，同时还可以模拟比赛的环境，培养那些计划参与 ACM/ICPC 赛事的队伍的团结合作能力。

1.2　在线评测系统在国内外的发展概况

在 ACM/ICPC 程序设计竞赛 30 多年的发展历程中，出现过很多优秀的 Online Judge 系统。

在国外，比较知名的 OJ 系统 SGU(Saratov State University Online Contester)、UVa Online Judge、SGU(Saratov State University Online Contester)等；国内比较知名的 OJ 系统有北京大学、天津大学、浙江大学、南开大学在线评测系统；同时，在 Google Code 等项目托管平台上还有诸如 HUSTOJ（华中科技大学)、BNUOJ（北京师范大学)、SOJ（国防科技大学）等开源的 OJ 系统。

此外，在 Google Code 上海有一个 FPS 项目（Free Problem Set)，旨在提供开源的评测题目和数据格式，并促进促进各大 OJ 对该格式的支持，方便共享。

武汉大学 ACM/ICPC 集训队员杨传辉、吴永坚、杨宝奎等集训队员也曾经开

发过一个在线评测系统（代号 Noah）并加以改进（代号 Oak），已经在武汉大学 ACM/ICPC 集训队服务器上运行数年。

1.3 课题的必要性及其意义

尽管已经有上述多个 OJ 系统，但是其中大部分是闭源、封闭的，无法获取其源代码进行修改扩充以满足现有需求，因此武汉大学 ACM/ICPC 集训队才于 2005 年推出自己的专有 OJ。

但是该系统存在几个问题：一，效率不足，无法承受每年武汉大学程序设计大赛预选赛的压力；二，由于时间较久，该系统部分源码已经遗失，无法修正存在的问题以及添加新功能；三，由于该系统没有应用沙箱技术运行用户代码，存在安全隐患，可能直接导致服务器被劫持。

此后，Google Code 上陆续出现了几个开源 OJ 系统，但是在架构设计上仍不够完善，且无法与 Oak 系统的数据格式兼容，因此，有必要重新开发一个经过合理设计的新 OJ 系统，从根本上解决上述的问题，为集训队提供一个稳定可用的学习环境；同时，将该系统经过简单的修改扩充后即可用于其他应用，诸如算法课程的作业平台，以及计算机学院本科生程序语言训练平台。

1.4 课题取得的成果

本课题所实现的在线评测系统（代号 Land）自有可用的原型系统起，就已经运行于武汉大学 ACM/ICPC 的服务器上，并随着开发的深入不断更新，目前已经可以为整个武汉大学 ACM/ICPC 集训队提供练习、比赛的平台。在结合其他高校的经验和教训基础上，对在线评测系统提出了更符合队伍训练及教学练习发展方向和更好提升用户体验的需求，并对核心功能做了深入的探讨并实现了需求中提出的所有核心功能及提升用户体验的绝大部分功能。

第 2 章　系统架构设计

2.1　系统功能需求

作为一个在线评测系统，核心的功能是要保证本系统能够在保证服务器安全的前提下，对用户提交的程序进行编译、运行，并评判最后的结果，最后将精准获取的运行耗费时间、内存及运行结果或错误信息返回给提交程序的用户。

为了方便用户的使用，本系统采用 B/S 架构，只要用户使用的是有网络接入的计算机，就可以通过浏览器进行访问。

根据该系统的具体情况，在设计上将其分为以下四个组成部分：

1. 网站系统（Web 端）
2. 评测核心（Judge）
3. 评测核心封装层（Judge Wrapper）
4. 监听守护进程（Daemon）

各个模块之间的关系如图 2.1 所示。

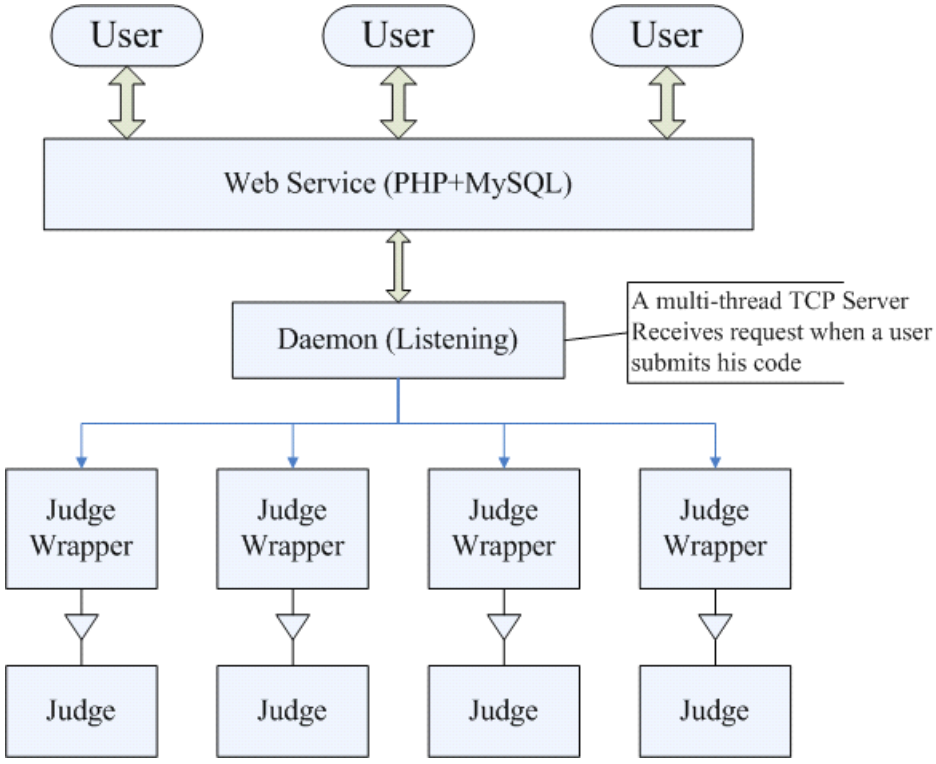

图 2.1　系统中各个模块之间的关系

在设计上，该项目期望将系统的几个主要组成模块充分解耦，一方面多个模块之间可以并行开发，另一方面，各个模块解耦后使得系统的修改和扩充更加容易，每个模块的可重用性也相应增强。例如，当其他学校需要实现一个功能界面完全不同的 OJ 系统或是作业平台时，可以直接采用本项目的评测核心，避免重复开发。

本章的其余部分将对各个模块进行综述，并在之后的章节中详细介绍各个模块的具体设计和实现。

2.2 网站系统（Web 端）

网站系统（Web 端）是整个系统与用户交互的部分，需要解决用户的注册、登录、提交代码以及查看结果等功能；同时管理员也需要使用 Web 端进行维护，包括但不限于增加新的题目、比赛，修改题目和比赛的信息，用户管理等功能。因此 Web 端的美观、易用等特性在其设计和实现中占有很重要的地位，对系统的可用性影响很大。

由于用户与 Web 端的交互通常是直接暴露在网络环境中的，还需要考虑相应的安全问题，比如用户的密码在传输和存储过程中可能存在的泄露，以及 CSRF 等攻击带来的系统安全问题。

此外，还需要充分考虑该系统今后的维护和扩充中可能存在的问题，因此，在该系统的设计中采用了新的架构和 MVC 设计模式也是非常有必要的。

2.3 评测核心（Judge）

评测核心（Judge）用于编译、运行和监控、比对用户程序的输出与标准输出之间的差异，反馈该代码的运行结果及相应的数据（运行时间、内存占用等）。根据 ACM/ICPC 赛事的相关规定，Judge 必须支持的语言包括 C、C++、Java；同时考虑到高中信息学竞赛选手通常使用 Pascal，因此在 Judge 的设计中一般也需要对 Pascal 的支持。

由于 Judge 引入了其他程序（编译器），并直接执行用户提交的代码，因此需要周全地考虑各种安全因素，以免用户提交的恶意代码对系统安全造成威胁。本系统的 Judge 利用 Linux 提供的 chroot、setuid、ptrace 等系统调用实现一个沙箱模

型，用于监控运行中的用户程序。

2.4 评测核心封装层（Judge Wrapper）

评测核心封装层（Judge Wrapper）是一个用于封装 Judge 的组件，通过屏蔽上层系统（Web 端）的具体特征，将 Judge 与其他模块彻底解耦。在本系统中，Judge Wrapper 的具体工作是调用 Judge 来配合 Web 端实现所需要功能。

在设计上将 Judge 独立出来，使得 Judge 针对的对象仅仅是具体的用户代码和输入输出数据有许多好处，不仅简化了 Judge 的开发、方便测试，使得 Judge 更具有通用性，并且，在 Web 端的功能有变动的时候，只需要修改 Judge Wrapper，而保持 Judge 不变，可以尽量减少修改 Judge 可能导致的安全风险。

2.5 监听守护进程（Daemon）

监听守护进程（Daemon）是一个多线程的 TCP Server。在设计上，Daemon 借鉴了操作系统中“中断”的思想：当用户提交一段代码以后，Web 端主动通知正在监听的 Daemon，然后由 Daemon 调用 Judge Wrapper，以触发对 Judge 的调用。尽管可以通过 Judge Wrapper 使用轮询的方式来检查是否有新提交的用户代码，但是效率较低，且会对数据库等造成而外的压力。

在 Daemon 的运行中需要考虑到多种因素，包括错误的请求、恶意请求、对错误的处理以保证其稳定运行；此外，由于可能同时有多个用户提交代码，需要保证不过度调用 Judge 而让系统无法承受，因此，还需要考虑到多个线程之间的同步等问题。

第 3 章　网站系统的设计与实现

如 2.2 节所述，Web 端需要实现的是易用、美观、安全、易维护、可扩展的用户界面。本章将展开阐述 Web 开发的背景、本项目的具体需求、数据库设计、架构设计与实现、安全相关等问题。

3.1　背景知识

Web 开发涉及的技术非常宽泛。本项目中采用的基于 LAMP(Linux、Apache、MySQL、PHP）架构进行开发，下面简要介绍本项目中需要用到的主要工具和语言和其他技术。

3.1.1　GNU/Linux（操作系统平台）

Linux 是一个基于 Posix 和 Unix 的多用户多任务支持多线程的类 Unix 操作系统，这个系统是由全世界各地的成千上万的程序员设计和实现的，其目的是建立不受任何商品化软件的版权制约的、全世界都能自由使用的 Unix 兼容产品。

Linux 以它的高效性和灵活性著称。它能够在 PC 计算机上实现全部 Unix 特性，具有多任务、多用户的能力。Linux 是在 GNU 公共许可权限下免费获得的，是一个符合 POSIX 标准的操作系统。Linux 操作系统软件包不仅包括完整的 Linux 操作系统，而且还包括了文本编辑器、高级语言编译器等应用软件。

由于 Linux 开放、免费的特性，有很多开源的程序和平台都是基于 Linux，或者对 Linux 提供了良好的支持，而 Apache、MySQL、PHP 等项目组合在 Linux 上表现尤其出色，因此特别适合用于 Web 开发。

3.1.2　Apache（Web 服务器）

Apache 是世界使用排名第一的 Web 服务器。它可以运行在几乎所有广泛使用的计算机平台上。

Apache 源于 NCSAhttpd 服务器，经过多次修改，成为世界上最流行的 Web 服务器软件之一。Apache 取自“a patchy server”的读音。意思是充满补丁的服务器，

因为它是自由软件，所以不断有人来为它开发新的功能，新的特性，修改原来的缺陷。Apache 的特点就是简单，速度快，性能稳定。

3.1.3 MySQL（数据库）

MySQL 是一个小型关系型数据库管理系统，开发者为瑞典 MySQL AB 公司。目前 MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。

MySQL 使用 C 和 C++编写，可移植性高，支持多种主流操作系统，为多种编程语言提供了 API，包括 C、C++、Eiffel、Java、Perl、PHP、Python、Ruby 和 Tcl 等；MySQL 支持多线程，充分利用 CPU 资源且实现了优化的 SQL 查询算法，有效地提高查询速度。更重要的是，实践检验 MySQL 可以处理拥有上千万条记录的大型数据库，能够支撑起 OJ 系统长期运行产生的大量用户数据。

3.1.4 PHP（服务器端动态脚本语言）

PHP，全称为 PHP Hypertext Preprocessor，超文本预处理器之意。PHP 是一种在服务器端执行的嵌入 HTML 文档的脚本语言，语言的风格类似 C 语言，现在被很多的网站编程人员广泛的运用。

PHP 独特的语法混合了 C、Java、Perl 以及 PHP 自创的新的语法。它可以比 CGI 或者 Perl 更快捷的执行动态网页。用 PHP 做出的动态网页与其他的编程语言相比，PHP 是将程序嵌入到 HTML 文档去执行，执行效率比完全生成 HTML 标记的 CGI 要高很多。PHP 执行引擎还会将用户经常访问的 PHP 程序的机器码缓存在内存中，其他用户再一次访问这个程序时就不需要重新编译程序了，只要直接执行内存中的代码就可以了。PHP 具有非常强大的功能，而且支持几乎所有流行的数据库以及操作系统。

使用 PHP 进行 Web 应用程序开发的优点异常丰富：开发效率高，函数语句简洁明了；输出控制灵活；可实现模板化，实现程序逻辑与用户界面分离；跨平台，可运行在 Win32 或 Unix/Linux/FreeBSD/OS2 等平台；与多个 Web 服务器兼容如 Apache、MS IIS、Netscape Server；完全支持面向对象开发，并向下兼容，支

持面向过程与面向对象两种风格开发；内置 Zend 加速引擎，性能稳定快速；内置函数丰富，几乎包含了 Web 开发的所有方面；组件化开发，提供 MySQL、Oracle、MS SQL 等多种数据库访问接口，支持 ODBC；支持正则表达式，内置 POSIX 与 Perl 兼容两类的正则表达式支持；开发成本低，开发工具多，且有众多使用 PHP 开发的源代码项目供我们参考和二次开发。

3.1.5 Javascript（客户端脚本语言）

Javascript 是一种基于对象和事件驱动并具有安全性能的脚本语言，它运行于客户端浏览器。在 HTML 基础上，使用 Javascript 可以开发交互式的 Web 网页(DHTML)，使得网页和用户之间实现了一种实时的、动态的，交互性的关系，使网页包含更多活跃的元素和更加精彩的内容。Javascript 使有规律的重复的 HTML 文段简化，减少下载时间。Javascript 能及时响应用户的操作，对提交表单做即时的检查。Javascript 短小精悍，又是在客户机上执行的，大大提高了网页的浏览速度和交互能力，在许多应用上，既提高了用户体验，又减轻了服务端的压力。

3.1.6 设计模式，MVC 设计模式

设计模式是一套被反复使用、多数有经验的开发者知晓、经过分类编目、对面向对象软件设计经验的总结。设计模式使开发设计人员可以更加简单方便地复用成功的设计与体系结构，做出有利于系统复用和扩展的选择，避免设计损害了系统的复用性和可扩展性。

MVC（Model View Controller）设计模式是 Xerox PARC 在八十年代为编程语言 Smalltalk－80 发明的一种软件设计模式，至今已被广泛使用。最近几年被推荐为 Sun 公司 J2EE 平台的设计模式，并且受到越来越多的使用 ColdFusion 和 PHP 的开发者的欢迎。它强制性的将应用程序的输入、处理和输出分开，使用 MVC 应用程序被分成三个核心部件：模型（Model）、视图（View）、控制器（Controller），让它们各自处理自己的任务。通过采用 MVC 设计模式可以实现低耦合性、高重用性和可适用性、较低的生命周期成本、快速的部署、可维护性、有利于软件工程化管理。

3.2 Web 端具体需求

作为一个在线评测系统的 Web 端，可以根据使用对象上分普通用户界面和管理界面成两大部分。

3.2.1 普通用户界面

一般用户获取的信息、交流和功能需求包括：首页（Homepage），题目（Problems），比赛（Contests），代码提交（Submit），实时状态（Status），用户排名（Ranklist），站内信（Mail），用户控制台（User），帮助（FAQ）。

1. 首页（Homepage）

首页包括欢迎信息, 最近的通知，最近比赛提醒，勤奋用户排行榜，友情链接等信息。同时，对于已经登录的用户，如果有未读的站内信，需要通过高亮显示站内信链接予以提示。

2. 题目（Problems）

题目包含题目列表，题目详情，题目统计信息等内容。

（1）题目列表：题目列表页每页显示 100 个题，每行一个题目的信息，信息包括用户是否通过此题、题目编号、该题通过率、该题提交次数和通过次数等信息。题目列表下方搜索框提供根据题号等条件的搜索。此外，考虑到用户需求，需要提供根据题目的通过率等信息对题目进行排序的功能。

（2）题目详情：题目详情页包含此题目的标题、描述、输入说明、输出说明、样例输入、样例输出，时间限制、内存限制、提交次数、通过次数、是否属于 Speicial Judge 题型（拥有不唯一的有效答案）等必要部分，描述中支持图片、特殊符号等内容的显示。每个题目还可以包含提示，来源，相关比赛三个非必要部分。

（3）题目统计信息：题目统计信息页面应包含某一题的提交信息统计表、通过的提交表，本题提交信息统计表包括尝试的用户数、通过的用户数、提交总次数、通过次数、格式错次数、超内存次数、超时次数、答案错次数、超输出次数、运行错次数、编译错次数。通过的提交表格式同实时状态信息，且只选通过的结果。同一用户多个通过的代码选成绩最好的；所有提交信息按成绩排序, 排序规则为以耗时为第一关键字递减、耗用内存为第二关键字递减。

3. 比赛（Contests）

比赛包含比赛列表、比赛详情、比赛排名、比赛统计信息、虚拟比赛列表、预定虚拟比赛、管理虚拟比赛等内容。

（1）比赛列表：分页显示所有的比赛信息；每行显示一场比赛，比赛信息包括比赛名称、开始时间、结束时间、比赛状态、比赛类型（公开比赛或私有比赛）等。

（2）比赛详情：主题为比赛题目列表，显示当前用户是否通过此题，题目编号，题目标题等。其他信息包括比赛开始时间、结束时间、当前时间、比赛状态等。

（3）比赛排名： 比赛排名包含一个表格, 每行分别是一个用户的信息, 包含排名（Rank）、用户名（username）、通过题数（Solved）、罚时（Time）、及各题的状态。

（4）比赛统计信息：比赛统计信息包含对用户和题目的统计信息。对用户的统计信息包括参加的用户数（即有提交的用户）、用户出题数的分布。对题目的信息包括每个题的各种结果数及总提交数的统计。

4. 实时状态（Status）

实时状态是一个独立的页面，通过列表的方式显示用户提交的代码的信息；同时提供服务给特定题目、比赛、用户等。提供服务给特定题目可以直接使用实时状态的搜索功能实现。实时状态表的每行应包含提交编号、提交者、提交题号、结果、内存使用、耗时、编程语言、代码长度、提交时间等。

5. 代码提交（Submit）

代码提交页面显示给用户一个提交表单，其中包括提交题号，编程语言，是否共享代码，源代码内容等项目。对于没有登录的用户提供匿名提交；在用户提交前对用户的代码进行检查，包括其代码长度以及语言类型是否匹配，减少用户错误选择语言的情况。

6. 用户排名（Ranklist）

用户排名页面每行包含一个用户的排名、用户名、昵称、用户通过题目数、用户提交次数和成功率等信息。排序的依据是通过题目数递减、提交次数递减、成功率递减。

7. 站内信（Mail）

站内信页面提供了系统内用户之间交流的方式。主要包括收件箱、发件箱、

邮件详情页、发信页面等。

（1）收件箱（Inbox）

收件箱分页显示用户收到的邮件，每行显示一封邮件的简要信息，包括邮件编号、发件人、邮件标题、发送时间以及相关操作按钮。未读邮件标题高亮显示。

（2）发件箱（Outbox）

发件箱使用同收件箱的方式分页显示用户发送的邮件。

（3）邮件详情（Detail）

邮件详情页显示邮件的具体内容，包括发件人，收件人，发送时间，标题和邮件正文。同时显示一个回复和删除按钮。

（4）发信页面（Send）

发信页面显示一个表单，包括收件人、标题和邮件正文。如果用户是点击邮件详情页面的回复按钮导航至发送页面，则自动填入收件人、标题，并截取部分该邮件正文以便识别会话。

8. 用户控制台（User）

用户控制台包含用户注册、用户登录、用户信息修改、用户信息详情页面。其中注册、登录、信息修改页面还需要考虑到用户密码在提交到服务器的传输过程中的安全性。

（1）用户注册：提供给用户一个表单，包括用户名、密码、昵称、学校、邮箱、是否共享代码等注册所需个人信息。

（2）用户登陆：用户名和密码，登陆后记录会话信息，退出后清除。

（3）用户信息修改：修改个人信息。

9．帮助（FAQ）

帮助页面给出使用本系统时的常见问题解答。

3.2.2 管理界面

管理员控制台的功能需求包括快速启动页（QuickLaunch）、题目控制台（Problems）、比赛控制台（Contests）、用户管理控制台（Users）、用户组管理控制台（Groups）、通知设置控制台（Notice）。由于管理员具有操作整个系统的权限，因此应当谨慎考虑各种安全问题。以下将分别说明上述页面。

1. 快速启动页（QuickLaunch）

快速启动页包括其他管理功能的入口，包括添加题目、管理题目相关数据文件、添加比赛、用户信息控制台、用户组管理、通知设置等功能。

2. 题目控制台（Problems）

题目控制台包括题目列表、查看题目详情、添加新题、修改题目详情、管理题目相关文件。为了避免重复开发，其中题目列表、查看题目详情等已有功能直接复用用户界面中的相关页面。添加/修改题目详情页面提供一个包含题目所有信息的表单共管理员填写。其中题目描述中需要支持富文本格式，包括超链接、图片、表格等，可采用 fckeditor 等所见即所得编辑器。管理题目相关文件页面包括输入输出数据文件的查看、添加和删除功能，以及 Special Judge 类型题目的 Special Judge 程序的查看、添加和修改功能。

3. 比赛控制台（Contests）

比赛控制台包括比赛列表、添加新比赛、修改已存在的比赛、查看比赛详情、管理比赛题目顺序等。比赛列表和比赛详情页面复用普通用户界面中的实现，并增加管理相关链接。添加/修改比赛页面提供一个包含比赛相关信息的表单供管理员填写。

4. 用户管理控制台（Users）

用户管理控制台允许管理员修改指定用户的信息，并指定用户组。

5. 用户组理控制台（Groups）

用户组管理控制台允许管理员查看现有用户组列表、添加/修改/删除用户组。

6. 通知设置控制台（Notice）

通知设置控制台允许管理员查看和修改系统通知。

3.3 数据库设计

数据库的设计是否合理，对系统开发、维护及性能的影响是尤为关键的。符合数据库设计 1NF、2NF、3NF、BCNF 等范式的设计对开发更有帮助；但是完全符合这些范式，效率又会偏低。因此，本系统在仔细分析 3.2 节的需求的基础上，结合数据库的设计规范，创建了题目（problems）、用户（users）、站内信（mails）、用户组（groups）、用户代码（sources）、管理员代码（admin_sources）、比赛（contests）、比赛题目信息（problem_at_contest）、比赛用户信息（user_at_cotnest）共九张数据

表来实现系统中的各种功能。数据库的具体结构见附录。本节将对数据库的具体设计作出详细的说明。

3.3.1 题目表

题目表（problems）中的每一行代表一个题目，包含了每个题目的基本信息，包括题目编号、描述、输入、输出、标准输入、标准输出、提示、来源、所属比赛编号、所属比赛中的题号、时间限制、内存限制、是否属于 Speicial Judge 类型、是否启用等字段。此外，还增设两个字段 accpeted 和 submitted，表示该题目通过人数和提交人数，这样虽然违反了数据库设计范式，但是在在每一次显示题目列表时就不需要反复查询用户曾经提交的代码的历史记录，极大地提高了效率。

3.3.2 用户表

用户表（users）中的每一行代表一个用户的基本信息，包括用户编号、用户名、加密后的密码、昵称、邮件、学校、注册时间、是否允许代码共享、所属组标号列表、是否启用等。如同 3.3.1 节 Problems 表的设计，在 users 表中增设几个字段保存提交次数、通过题数和通过的题目列表，可以极大地提高在用户详情页、用户排名页和其他地方的查询效率。

由于系统管理和匿名用户的需要，表中默认添加一个 root 管理员用户和一个匿名用户。

3.3.3 用户组表

用户组表（groups）每一行存储一个用户组的基本信息，包括组编号、组名称、各种权限（如管理权限、查看代码、参加私有比赛等）。通过增设用户组功能，可以更方便地进行用户管理。

表中默认添加一个管理员组。

3.3.4 邮件表

邮件表（mails）的每一行包含一封邮件的信息，包括邮件编号、发信人、收信人、发送时间、标题、正文、收件人已读、收件人删除标志、发件人删除标志。

3.3.5 用户代码、管理员代码表

用户代码（sources）、管理员代码（admin_sources）两个表的每行存储一个用户/管理员提交的代码的信息，包括代码编号、题目编号、用户编号、用户名、比赛编号、源代码、代码长度、提交时间、提交 IP、语言类型、是否共享、评测时间、内存占用、时间占用、结果、额外的信息（如编译错误等）。

用户提交的代码参与用户排名和比赛排名；管理员提交的代码仅仅用于测试题目和测试数据是否正确，因此需要分开存储。

3.3.6 比赛表

比赛表（contests）每一行包括一场比赛的基本信息，包括比赛编号、比赛名称、比赛描述、是否私有、开始时间、结束时间、是否启用等。

3.3.7 比赛题目信息表

比赛用户信息表（problem_at_contest）存储了比赛题目的统计信息。其主键是比赛编号和题目编号，每一行中存储的是一场比赛里，对某一个题目提交代码的统计数据，包括通过、格式错误、编译错误、答案错误、运行时错误、超时、占用过多内存、输出过多数据和总提交数等。

对于某一场比赛，通过查阅该表的相关信息，就可以很高效地给出一个比赛所有题目的统计数据表格，避免了大量查询 3.1.5 的用户代码表给数据库带来的压力。

3.3.8 比赛用户信息表

比赛用户信息表（user_at_contest）存放了参加比赛的用户的统计信息，其主键是比赛编号和用户名。该表的每一行存放参加了某一场比赛的某个用户的信息，包括用户编号、用户名、比赛编号、通过题数、罚时、以及一个额外的字符串字段以 json 格式存储了用户对比赛中某体的提交情况。

通过这种方式使得在查看比赛的用户排名页时可以很高效地给出统计表格，避免了大量查询 3.1.5 用户代码表给数据库带来的压力。

3.4 架构设计与实现

如 3.1.6 节中对 MVC 的介绍，在中大型 Web 系统的设计中，MVC 设计模式的应用可以大幅度降低开发的复杂性，提高开发的效率，且易于让处理逻辑开发人员和显示逻辑开发人员分工合作。因此，在 Land 的 Web 端的设计中，MVC 设计模式是必不可少的。

为了实现 MVC 设计模式并将其的好处最大化，一个合理的代码框架是非常重要的。在 Web 端的设计中，首先借鉴了 REST 设计原则，将对所有资源的请求用 Apache 的 mod_rewrite 重写到同一个文件（index.php）。在 index.php 中固化一个完整的执行路线：完成必要的初始化工作；根据用户的请求载入相应的执行模块对请求进行处理；获得模块执行的结果，并载入指定的显示模块渲染页面返回给用户。此外，当请求过程中出错，也提供统一的错误处理模板，完成必要的收尾工作并给用户友好的提示信息。当这样一个框架设计好后，MVC 设计模式的目标也就达到了。

该流程图如图 3.1 所示。

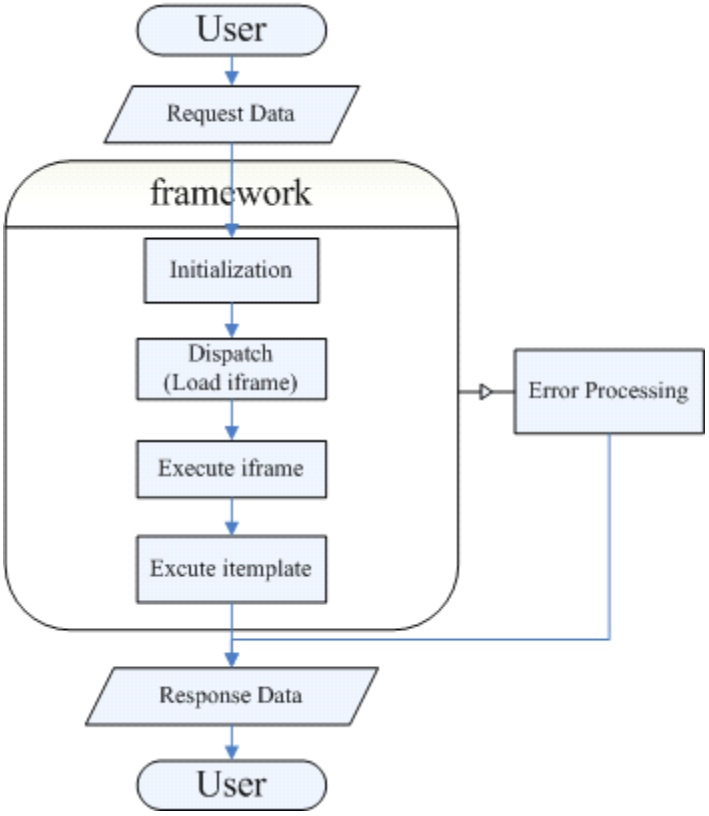

图 3.1　Web 端总体框架流程图

本节的其余内容将更详细地介绍设计中的细节。

3.4.1 重写请求

重写请求的实现非常简单，根据 Apache 的配置文件写法，只需要在 Web 目录（Land/web/index）下写入一个文件名为“.htaccess”的配置文件，内容为：

```
RewriteEngine On
RewriteBase /land/
RewriteCond %{REQUEST_FILENAME} -f
RewriteRule ^.*$ - [QSA,L]
RewriteRule ^.*$ index.php [QSA,L]
```

该配置文件的含义是，

1. 启用 URL 重写引擎
2. 需要重写的 URL 前缀是 /land/
3. 如果请求的文件是存在的，重写规则为“保持不变”，重写结束
4. 将其他所有请求重写到 index.php，重写结束

通过上述重写规则，保证了静态文件（如 js、css、图片等）的快速访问，同时将其他请求重写到了 index.php，控制了请求的处理路线。

3.4.2 初始化

初始化过程中完成了每次请求过程中必然要完成的一部分工作，主要包含以下几个方面：

1. 包含必要的库文件、配置等。
2. 设置错误处理句柄，可以将 PHP 的运行时错误记录到日志文件中。
3. 打开日志文件。通过将运行时状态输出到日志文件，可以非常方便地实现快速定位错误位置和原因；同时也为日后的分析和统计带来了很大的好处。
4. 初始化请求数据。包括获取 HTTP 请求的方式、请求的 URL，处理 GET 或 POST 以及 COOKIE 数组，获取用户的 IP 地址，以及生成该用户唯一的日志编号，增加固定的日志信息等。

通过框架统一完成初始化工作，使得代码量大量减少，尤其是冗余代码的减少，

能够使得模块的开放变得更加简单、快捷。

3.4.3 载入执行模块

执行模块就是 MVC 设计模式中对应的 Controller 模块。此阶段的任务是根据预先设定的分发规则对用户请求的 URL 进行处理，获得相应的模块。载入该模块对用户的请求进行处理，指定相应的渲染模块，并返回相应的处理数据。

为了使框架能够调用模块，首先应当为模块和框架之间的交互定义统一的接口。本系统中进行如下设计：

```
interface iframe {
    public function pre_process();
    public function process();
    public function post_process();
    public function display();
    public function err_handler();
}
```

定义了一个接口“iframe”，接口中包含 pre_process、process、post_process 和 display 共 4 个用于正常执行流程的方法；err_handler 方法用于错误处理。

然后实现一个工厂函数（Factory，根据不同的输 入数据创建不同的对象），并在该函数内完成 iframe 提供接口的调用，代码如下：

```
class cframe_loader {
    public static $c;
    public static function run($class_name) {
        self::$c        = new $class_name();
        try {
            if (false == self::$c->pre_process())
                throw new Exception_frame("pre_process");
            if (false == self::$c->process())
                throw new Exception_frame("process");
            if (false == self::$c->post_process())
                throw new Exception_frame("post_process");
            if (false == self::$c->display())
                throw new Exception_frame("display");
            return self::$c;
        } catch(Exception_frame $e) {
            if (false == self::$c->err_handler()) throw $e;
        }
    }
}
```

只要模块的类实现了 iframe 接口，或者继承于是先了 iframe 接口的类，那么就可以通过 cframe_loader::run()来实例化并执行这个类的相应代码。上述框架利用了 PHP 的反射机制以及面向对象程序设计思想中的多态来实现，使得代码更加清晰、容易编写，且便于日后维护修改。

在具体的实现中，模块的类一般继承于 cframe 这个类，这个类在 pre_process 函数中做了一些额外的工作，包括检查用户是否已经登录，获取登录用户的相关信息等。此外，类 acframe 继承于 cframe 类，通过重载 pre_process 函数，额外检查了当前用户是否具有管理员权限，适用于系统中的管理页面。通过这样的方式，大量减少了开发时的编码量并降低了相应的风险。

3.4.4 载入渲染模块

渲染模块对应于 MVC 设计模式中的 View 模块。执行模块（Controller）在处理完用户的请求后会指定相应的渲染模块并给出所需的数据（实际上是通过调用系统提供的 Response 组件的相应方法），框架获取到相应的信息后即载入该渲染模块进行执行，生成页面返回给用户。

在具体的实现上，类似于 iframe/cframe/cframe_loader 的关系，创建了 itemplate 接口、实现该接口的 ctemplate 类和该类的工厂方法（ctemplate_run）。

itemplate 接口：包含一个 display 方法，传入数组 p，生成最终结果并输出。

```
interface itemplate {
    public function display($p);
}
```

ctemplate 类不仅实现 itemplate 接口，还重载了构造函数和析构函数用于载入固定的页首和页脚，较适合于普通页面的载入，省却了重复指定页首和页脚的麻烦，简化了页面的开发。

ctemplate_run 类包含了一个静态方法 run，根据输入参数给出的文件名和类名，载入包含相应类的文件并实例化该类，获得一个 itemplate 对象，将执行模块给出的数组传入，执行该对象的 display 方法，显示具体的页面。

通过这样的设计，使得在开发显示逻辑的时候，只要简单地继承于 ctemplate 类，然后在执行模块中指定该类的文件名即可，非常方便。

3.4.5 错误处理

在 index.php 的代码框架中，使用了 try-catch 结构，只要在正常的执行流程中抛出一个未被捕获的一场，就会被框架中的 catch 语句捕获，通过 LOG 记录该异常的错误信息，并向用户显示一个友好的错误页面。

3.5 安全相关

由于网站是直接与用户打交道，接受用户的输入，因此在 Web 开发中，有非常多的因素影响系统的安全和稳定。

3.5.1 用户密码安全

保证用户密码安全，主要包括两个方面：在服务器中存储的密码安全，以及在传输过程中的密码安全。

1. 在服务器中存储的密码安全。

由于服务器管理员可以查询数据库中的数据，同时也要考虑到数据库因误操作和服务器收到攻击等原因导致的非正常外泄，因此在数据库中存储的密码不能是明文。在本系统的设计中，采用 MD5 哈希算法对用户密码进行哈希，由于哈希算法的不可逆性，可以保证即使存在恶意管理员或者数据库外泄的情况，用户密码也不会被破解。

2. 在传输过程中的密码安全。

出于成本等原因考虑，大多数 HTTP 服务器并没有向证书管理机构（CA）申请证书。因此，在用户访问服务器的时候，所有的数据都是以明文的方式进行传输，攻击者只要能够在传输途径中进行监听，就有可能获得传输过程中的数据。

在本系统的设计中，采用了 javascript 在客户端对用户输入的密码（Password）进行 MD5 哈希以后再传输的方法，这样可以保证即使攻击者截获传输数据，也无法获得密码明文。但是，仅仅这样的设计是不够的：攻击者截获哈希后的密码，仍然可以向服务器发起伪造的请求，将用户密码修改，从而控制用户帐户。因此，还需要增加一个额外的、每次请求都变化的、且只能使用一次的随机数（Seed），以如下公式生成一个签名（Signature）：

$$Signature = MD5(MD5(Password) + Seed)$$

在与服务器交互的过程中，只传输 Signature 和 Seed，服务器端收到 Signature 和 Seed 以后，首先检查 Seed 与服务器端存储的 Seed 是否一致，保证每个 Seed 只使用一次。然后将数据库中存储的用户密码哈希值代入上式，将计算出的值与该 Signature 进行比较，如果两个值一致，则表示用户输入的是正确的密码。同时，应当立即令该 Seed 失效，使得攻击者即使截获了本次传输的数据，也无法再次利用该数据。

3.5.2 数据库用户名和密码的安全

由于向 MySQL 服务器发起连接时，在 PHP 源码中必须提供数据库密码明文，

所以 PHP 的代码安全就非常重要。由于 HTTP 服务器可能存在安全漏洞（如微软公司的 IIS 服务器曾经出现过可以让用户下载任意文件的漏洞），如果存放数据库密码的代码被恶意用户下载，对 MySQL 服务器的安全就造成了威胁。因此，本系统中采用了将除了 index.php 之外的所有代码都放置在 index 文件夹之外的方式，保证即使 HTTP 服务器出现问题，用户也无法下载到包含敏感信息的代码，从而保证数据库的安全。

3.5.3 SQL 注入攻击

在 Web 开发中，SQL 注入攻击是非常常见的攻击方式。由于进行请求时经常需要在 SQL 语句中加入用户输入的数据，如果没有严格的检查，恶意用户可以通过构造出恶意的数据，使得该 SQL 的执行超出了预期的结果。例如，如下 SQL 语句是常见的用于验证用户登录请求是否正确，其中$username 和$password 是用户输入的用户名和密码。如果用户构造的用户名是“1' OR 1 OR '”，这样就可使得该 WHERE 子句的结果始终为真。

```
SELECT COUNT(*) FROM `users`
    WHERE `username`='$username' AND `password`='$password'
```

因此，在处理每一个包含用户输入数据的 SQL 语句时，必须进行 SQL 转码，过滤用户的恶意输入。

3.5.4 CSRF 攻击

CSRF 是伪造客户端请求的一种攻击，英文全称是 Cross Site Request Forgery，字面上的意思是跨站点伪造请求。CSRF 攻击强迫受害者的浏览器向一个易受攻击的 Web 应用程序发送请求，最后达到攻击者所需要的操作行为。

一个典型的 CSRF 攻击是伪造一个包含自动提交表单的页面，欺骗系统管理员访问，从而能够以管理员权限进行一些非授权的操作，例如修改用户密码、权限等。

因此，在本系统的设计中，系统管理员的每一个可能影响系统安全的操作，都使用了类似于 3.5.1 中介绍的 Seed 的方式进行验证，再请求前的表单中加入一

个随机的、只能使用一次的 Seed，并在提及时进行验证，使得伪造的请求无法通过验证。

3.5.5 XSS 攻击

XSS 又叫 CSS (Cross Site Script) ，跨站脚本攻击。它指的是恶意攻击者往 Web 页面里插入恶意 html 代码，当用户浏览该页之时，嵌入其中 Web 里面的 html 代码会被执行，从而达到恶意用户的特殊目的。

一个典型的 XSS 攻击会在其页面中通过<script src="">标签嵌入用户可能访问的站点的一段代码，获得用户的 SessionID，从而可以伪造用户访问该站点，侵入用户帐户。

本系统的设计中，ajax 请求采用的都是 POST 方式，从而避免了此类攻击可能造成的威胁。

3.6 Web 端设计小结

Web 端的开发是本系统中代码量最大的部分。虽然每一个子模块的复杂性都不高，但是 Web 系统的设计——也就是如何让各个模块有机地结合起来成为一个整体、尽量降低在系统复杂度增加过程中其余各个问题带来的影响、并为将来的维护和扩展留下足够的空间，则是一件需要反复斟酌的问题。总体而言，本系统在 Web 端的设计较好地达到了预期的目标，但是鉴于个人在设计上的经验不够充分，在具体的实现上难免仍然会存在一些瑕疵。

第 4 章　评测核心的设计与实现

作为在线评测系统中最核心的部分、尤其是需要监控用户运行，需要涉及到很多相关技术，特别是与系统底层关系密切的技术。本章将对评测核心需要涉及的技术和其他背景知识给出简明、必要的描述，然后详细阐述评测核心中的初始化、编译、运行和判断结果等过程的具体设计和实现。

4.1　背景知识

4.1.1　编译器

评测核心获得用户提交的源码以后，需要先对其进行编译生成可执行文件。如 2.3 节所述，本系统支持四种语言，分别是 C、C++、Java 和 Pascal，根据 ACM/ICPC 赛事的惯例，我们选用了 GCC、G++、Sun-JDK 和 FreePascal 作为所需支持语言的编译器。

1. GCC（GNU Compiler Collection），是由 Stallman 所领导的 GNU 项目下的编译器，可以编译的语言包括 C, C++, Objective-C, Fortran, Java,Ada 等。一般而言，仅安装基本的 GCC 套件仅支持编译 C 语言。

2. G++是 GNU C++的惯用简称，是 GCC 项目的一部分，用于编译 C++。

3. Sun-JDK（Java Development Kit）是 Sun Microsystems 公司为 Java 开发人员发布的产品。Sun-JDK 中包含 JRE（Java Runtime Environment）、Java 编译器、调试器、类库等众多用于开发调试的工具。

4. FreePascal 是针对 Pascal 和 Object Pascal 的一组免费、自由、可移植、开源的编译器。

4.1.2　系统调用

系统调用是程序向操作系统内核请求服务的接口。在大多数现代操作系统中，内核运行于高特权级，拥有整个硬件平台的控制权限；用户程序则在内核的监控下运行于低特权级，以保证整个系统的安全和稳定。当用户程序需要完成一些需

要高特权的操作时，即通过系统调用向内核请求服务，由内核来完成相应操作。

在 Linux 下，系统调用的实现通常是用户程序通过触发 80 号软中断或者执行 SYSCALL/SYSENTER 等平台相关的 CPU 指令陷入到内核中，内核通过寄存器获取用户程序的输入，在通过严格的检查后执行相应操作。

在本系统中涉及的主要系统调用包括 fork、setitimer、execve、wait、ptrace、setrlimit、chroot、setuid 等，下面将分别简要介绍各系统调用，更详细的内容可参见 posix 规范或 Linux 系统的联机帮助（man page）。

1. fork 系统调用通过复制调用进程的上下文来创建一个新进程。C 库中的 fork 函数是对该系统调用的封装，该函数被调用一次，却返回两次：向父进程返回子进程的 pid，向子进程中返回 0。由于 fork 函数并没有改变进程的上下文，因此父子进程都沿着原有的程序继续执行，只能由返回的 pid 来识别。

在 Linux 下，fork 系统调用采用了 COW(Copy-On-Write)机制，即仅仅复制父进程的页表、创建新的 task structure，使得 fork 的执行开销更小。

2. setitimer 系统调用用于设置定时器。Linux 系统为每一个进程安置了 3 个可用的定时器（真实时间、进程执行用户时间、进程执行总时间）。此系统调用可以取得定时器的当前状态、设置新的时间等。

3. execve 系统调用用于载入一个新的可执行程序，替换当前进程的地址空间。C 库中对该系统调用的封装有 execl、execle、execlp、execv、execvp 等多种形式。该系统调用成功后不会返回，将直接执行新的程序。

4. wait 系统调用允许父进程阻塞，直到子进程发生一些事件。本系统多次使用 wait 系统调用，用于同步父进程和子进程，例如等待编译器运行结束、等待被监控用户程序的系统调用或结束等。

5. ptrace 系统调用是一个可以使父进程在用户层拦截和修改系统调用的函数，可以监控和控制其他进程，该函数还能够改变子进程中的寄存器和内核映像，因而可以实现断点调试和系统调用的跟踪。

6. setrlimit 系统调用可以改变进程的资源限制。Linux 支持的资源限制非常丰富，在本系统中使用到的包括 CPU 时间、栈空间大小、和输出文件大小。

7. chroot 系统调用使得调用进程的将某一目录当作其根目录，以此来限制该进程及其子进程对该目录外的文件的访问。

8. setuid 系统调用允许进程改变其有效用户 ID。当调用进程的有效用户 ID

是 root，那么其真实用户 ID 和保存的 set-user-ID 也会跟着改变。通过此系统调用可以改变进程的身份，由此来限制进程的权限。

4.1.3 文件权限，SUID 位

在支持 POSIX 规范的操作系统下，每个文件都拥有 3 组权限：所有者(user)、所有者所在组(group)、其他用户(other)。每个组的权限包含读（R）、写（W）、执行（X）三种。因此，一般采用 3 个 3 位的 8 进制数来表示文件权限，如 0755 表示所有者权限是可读、可写、可执行，所有者所在组和其他用户的权限是可读、可执行，不可写。

在实践中发现，类似 ping、traceroute 等程序在执行时（创建 icmp 包）需要 root 权限，但是普通用户也有需求，因此 POSIX 规范中增加了 SUID、SGID 等位，允许设置了 SUID/SGID 位的可执行程序在运行时的有效用户 ID（euid）是其所有者。

在本项目中，由于 chroot 和 setuid 系统调用需要 root 权限，因此，可以通过以下命令可以 judge 可执行程序的 SUID 位，使得 judge 即使是由普通用户执行，也可以正常运行：

```
$ sudo chmod 4755 judge_all.exe
```

4.1.4 沙箱环境

沙箱指的是一种安全的环境。在计算机相关领域，沙箱环境，指的是为因动态修改而引起风险的应用程序创建的安全环境，通过提供一个限制权限的环境来运行程序或程序的某一部分，从而将威胁限制在可控的范围内。

在本系统中，用户提交的代码可能存在恶意，对其编译后的程序需要运行在服务器上，因此必须对其进行严密的监控，防止恶意代码带来的危害。在本系统支持的四种语言中，C、C++、Pascal 三者经过编译后产生本地代码，可以通过 4.1.3 中介绍的 Linux 系统调用（主要是 fork、execve、chroot、setuid 和 ptrace）来创建一个沙箱，限制恶意代码访问文件、控制系统的权限，尤其是通过 ptrace 来限制可使用的系统调用，在很大程度上保证了系统的安全。对于 Java 程序，由于是运行在 JVM 之上，如果直接采用上述沙箱环境，则会导致 JVM 无法正常运行。因

此，本系统采用了 JVM 的 Security Manager 和 ptrace 系统调用相结合的方式为 Java 程序创建了一个不同的沙箱环境。

由 fork、execve 等系统调用构成的沙箱模型流程大致如图 4.1 所示。

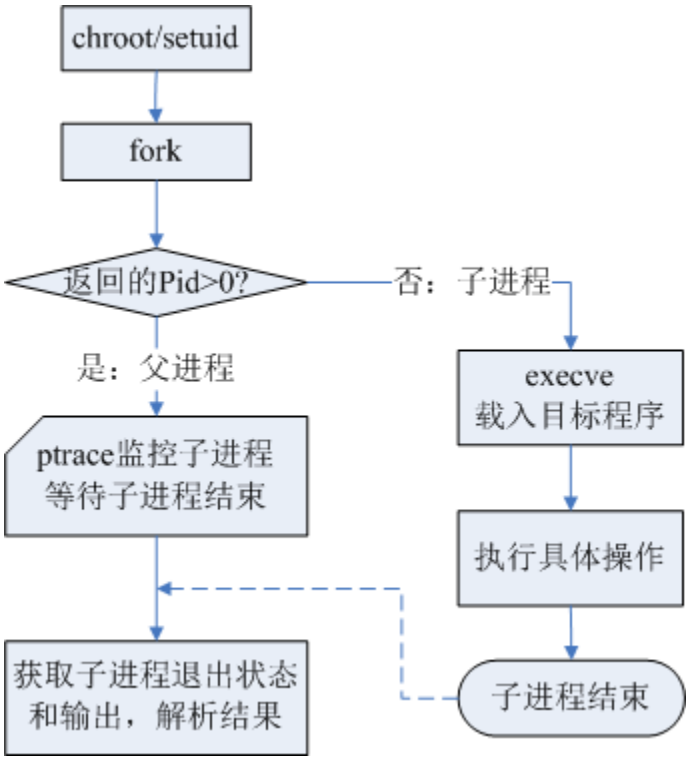

图 4.1　由系统调用构成的沙箱模型流程

4.1.5　Linux 的内存管理和进程的内存占用

在 32 位的系统中，硬件支持的内存大小上限是 4G，因此 Linux 为每一个进程提供的虚拟内存空间也是 4G。其中 Linux 内核占据其中 3～4G 之间的空间，进程可以使用的地址空间为 0～3G。

在进程运行的过程中，如果需要额外的可用空间，一般是通过 brk/sbrk 系统调用来扩充其地址空间。但是，分配到的空间绝大部分并不会立刻投入使用，出于效率的考虑，在 brk/sbrk 系统调用执行过程中并不真正分配地址空间，而是仅仅初始化页表。当进程需要读写其中的某一地址时，由于该地址所在的页面并不真正存在于物理内存中，会触发缺页中断陷入内核，此时内核才分配一个页面给该进程。

因此，进程的地址空间并不能真实地反应出进程消耗的内存；而进程触发的缺页中断数正好准确地描述出了进程该进程自运行起真正分配的页面数量，进而

可以计算出该进程的真实内存占用数。

4.1.6 在线评测系统对用户程序输出的判断标准

一般情况下，在线评测系统通过对比用户程序输出和对应的标准输出两个文本文件的相似度来判断用户程序是否给出了正确的输出，即得到正确结果。为简化处理流程及增大对用户的挑战性，在评测系统中将相似度只分成三个层次，即完全一致（Accepted），格式错误（Presentation Error）和不一致（Wrong Answer）。完全一致要求两个文本文件内所有字符均一致。格式错误是指如果两个文件不完全一致，但是在去除两个文件的所有分隔符（如空格，制表符及换行符等）后所有字符均一致。如果两个文本文件，不完全一致且不是格式错误，均判为不一致。在评测内核中，只有完全一致的比较结果被认为是正确的结果。

在 ACM/ICPC 中有一类题目，根据其题意，符合条件的输出不止一组。这一类题目被称为“Special Judge”题目，即需要与上述判断标准不同的特殊判断方法。由于每个 Special Judge 题目的题意不同，无法统一判断标准，因此每一题需要提供一个额外的程序用于判断用户输出是否符合题意。

4.2 初始化

在本系统的设计中，评测核心与其他模块充分独立，因此需要约定一个详细的接口来完成与其他模块（核心封装层，将在第 5 章介绍）之间的交互，具体的接口细节比较繁琐，限于篇幅不再本文中详细描述，可参见评测核心代码目录下的说明文件（readme.txt）。

评测核心的初始化工作包括很多内容，最重要的部分是获取并组织好本次运行所需的参数；还有其他非常重要的操作需要执行，按顺序分别介绍如下：

1. 载入 Judge 的配置文件。读取同一目录下的 config.ini，解析出日志文件路径、SPJ 程序的执行时间限制等配置参数。
2. 打开日志文件，为此后写日志做好准备。
3. 检查运行时环境，由于 chroot 和 setuid 函数要求 root 权限，需要检查有效用户 ID（euid）是否是 root。
4. 根据该接口定义的规范，获取并组织好本次运行所需的参数。包括用户程

序源文件路径、所用语言、题目编号、数据目录、临时目录、时间限制、内存限制、输出限制、是否 Special Judge 等。

5. 设置 Judge 的执行时间限制。因为系统资源有限，同时可以执行的 Judge 数量需要有一定的限制，如果一个 Judge 由于为止原因（如代码中存在的错误）卡死，会影响后续评测的进行；在极端情况下连续多个 Judge 卡死，就会造成评测的完全停止，需要人工干预才能继续进行。

6. 设置相关的信号处理句柄，使得相关信号（SIGALRM）能够被捕获，完成必要的清理工作。

4.3 安全编译

简而言之，编译阶段就是使用编译其将用户提交的代码编译成可执行程序。但是由于编译器属于外部程序，可能本身存在问题；还有可能因代码出现编译错误而终止，因此，在编译阶段也需要考虑多方面的因素，来保证 Judge 的稳定执行。

由于使用了编译器，实际上就是引进了外部程序，同时也引进了不可控的因素。例如一，个使用模板元编程的 C++程序，可能因为（故意）漏掉终止递归的特例化模板代码而导致 G++编译时出现栈溢出而崩溃的情况；某些版本的编译器在处理特定代码的时候可能由于自身存在的 BUG 而非正常终止；由于用户提交的代码可能不符合语法规范，会出现编译错误。

因此，本系统采用了一个简单的沙箱环境（流程同图 4.1）用于编译，由 fork 产生的子进程调用编译器编译用户代码，并给出编译时间限制，保证编译的正常执行和错误检测。

4.4 运行和监控

运行和监控是评测核心中最重要的部分。由于用户代码编译后的程序直接运行于服务器上，引入的不确定因素非常多，因此，需要非常严格的沙箱环境，使得核心能够安全、平稳地运行。

图 4.1 给出了此环节的大体流程，但是在此环节中具体的沙箱环境有更加严格的要求。

4.4.1 输入输出

由于禁止用户程序开启文件，需要通过 freopen 函数来重定向用户程序的标准输入（stdin）、标准输出（stdout）、标准错误输出（stderr）三个流。

4.4.2 权限

通过 chroot 系统调用限制用户进程可访问的文件路径集合，通过 setuid 系统调用限制用户进程的权限（设置为持有最低权限的用户 ID，本系统采用了 nobody 这个操作系统内置的、权限最低的用户）。

4.4.3 资源限制

需要限制用户程序的各种资源使用，包括时间（CPU 资源）、内存资源、栈空间和输出文件大小。这些资源限制通过 setrlimit 和 setitimer 等系统调用结合来完成。

4.4.4 运行时检测

这是最重要、最复杂的运行时监控机制。本系统中通过 ptrace 系统调用来监控用户进程。每当用户进程出发一个系统调用，或者是收到某些信号的时候，用户进程会将控制权暂时移交给监控核心。监控核心将严格监控用户进程的各项参数，以保证其不执行危害系统安全的操作：

1. 检查程序是否正常退出。对于正常退出的程序，进入结果判断环节。

2. 检查程序是否收到了非正常的信号，包括 SIGALRM、SIGVTALRM、SIGXCPU、SIGXFSZ、SIGSEGV、SIGFPE、SIGBUS、SIGABRT 等信号。此类信号表示进程使用的资源超出了其限制或是进程执行了非法操作导致错误，需要结束进程。

3. 通过进程的缺页中断次数计算进程使用的内存资源是否超出限制。

4. 检查进程的系统调用是否合法。合法的系统调用编号保存在数组 RF_table 中。由于语言之间的差异，对 C、C++、Java、Pascal 分别有一个不同的规则定义合法的系统调用编号，这些系统调用编号能够保证，用户进程完成题目所必须的

操作能够不受干扰地进行。

在以上四项检测都不存在问题的情况下，用户进程才能继续执行。

4.5 判断结果

由于题目根据其合法结果数量可以分成普通类型和 Special Judge 类型，因此针对不同的类型，需要分别对其进行判断。

4.5.1 普通类型

普通类型的题目，其合法结果唯一，因此只需要简单判断用户程序与标准答案的一致性即可。如 4.1.6 所述，本系统在完全一致（Accepted）和错误（Wrong Answer）之间加入了格式错误（Presentation Error）这一标准，因此在对比时需要增加一些额外的机制来进行判断。

4.5.2 Special Judge 类型

由于每个 Special Judge 题目的题意不同，无法统一判断标准，因此每一题需要提供一个额外的程序（SPJ 程序）用于判断用户输出是否符合题意。在本系统中，约定了 SPJ 程序的名称以及数据的传递方式，包括标准输出、用户输出，以及 SPJ 程序的数据，由评测核心调用 SPJ 程序来完成最后的评判过程。

4.6 评测核心设计小结

评测核心的健壮性直接关系到在线评测系统的可用性。因此在本系统的开发和测试中，对评测核心倾注了最多的工作。但是，由于在线评测系统本身运行方式的局限，始终无法避免 DoS（Deny of Service）类的攻击。尽管如此，本系统中的评测核心被设计为完全独立，使得存在将评测工作扩展到多台服务器成为可能且容易实现，至少能够在一定程度上缓解此矛盾。

由于时间和精力上的限制，虽然此评测核心已经完全可用，但是仍有一些计划中的特性未能实现，包括对更多编程语言种类的支持，以及更安全地监控 SPJ 程序，以及上述的分布式 Judge 等。

第 5 章　评测核心封装层的设计与实现

如前所述，评测核心被设计成完全独立、便于扩展，因此，需要对其进行封装，以实现具体的业务需要。

5.1　背景知识

5.1.1　适配器模式

在计算机编程中，适配器模式（有时候也称包装样式或者包装）将一个类的接口适配成用户所期待的。一个适配允许通常因为接口不兼容而不能在一起工作的类工作在一起，做法是将类自己的接口包裹在一个已存在的类中。适配器模式的宗旨是保留现有类所提供的服务，向客户提供接口，以满足客户的期望。

5.1.2　命令行界面和脚本

命令行界面（CLI，Command Line Interface 的缩写），指的是为了执行某些任务，与计算机操作系统或者软件通过输入指令进行交互的机制。纯文本界面与带菜单且支持鼠标的图形用户界面（GUI）、带菜单的文本用户界面（TUI）相对。通过 CLI 操作计算机执行某个具体任务的方式被称为“输入”指令：系统等待用户提交文本命令并按下回车键，命令行解释器则负责接收、解析和执行请求的指令。执行完毕后，这条命令通常会通过文本的方式将执行的结果返回给用户。

将一系列命令组合成一个文件，并批量输入给命令行解释器让其执行，则此文件被称为脚本（Script）程序。

PHP 面向开发者提供了 CLI 界面，且支持脚本处理。

5.2　封装层的设计

在本系统的设计中，对 5.1.1 节提到的适配器模式的含义进行了适当的扩展，评测核心封装层（Judge Wrapper）即是评测核心相对于具体业务（不同的评测系统）的适配器。这样的设计与多条 Unix 哲学契合（程序应该只关注一个目标，并

尽可能把它做好；让程序能够互相协同工作），带来了诸多的好处，包括：

1. 评测核心可以仅仅关注评测相关工作。现有的开源在线评测系统的评测核心通常都包括了数据库操作，这使得评测核心的开发和测试都非常复杂。采用本系统的设计，使得在评测核心的开发和测试可以完全独立地进行。

2. 评测核心的可复用性大大增强。由于核心与具体系统的业务无关性，使得任意一个系统都可以为其编写一个特定的 wrapper，实现所需的功能。截至本文撰写时，浙江传媒学院已有一个正在开发的在线评测系统使用了本系统的评测核心。

3. 由于封装层（Wrapper）的工作是通过调用评测核心实现的功能来完成业务相关的工作，对其本身的执行效率要求不高，因此可以采用各种语言，尤其是方便开发和测试的脚本语言来编写。在本系统中，具体而言就是更新在线评测系统的数据库，不涉及大量的计算等工作。因此，采用 PHP 来开发封装层能够直接使用 PHP 和 Web 端已经实现的库，方便开发且易于测试。

4. 由于封装层和评测核心之间的松耦合，使得可以很容易地在二者之间加入一个分布式层，实现分布式评测。

5.3 封装层的实现

根据本系统 Web 端数据库的设计，封装层的工作主要分为几个阶段，如图 5.1 所示。

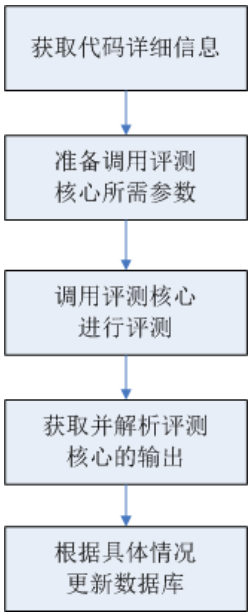

图 5.1　评测核心封装层工作阶段示意图

5.3.1 获取代码详细信息

每当用户提交一份新的代码，Wrapper 即会被调用，从数据库中获取代码的详细信息。在此阶段需要关注的信息主要包括：

1. 用户提交的代码本身及其关键属性（如语言类型）。这是一次评测过程的核心对象。

2. 此代码的提交用户类型，是普通用户还是管理员。在 3.3.5 节的代码表设计中提到，普通用户与管理员用户提交的代码使用了不同的数据表，因此需要予以区分。

3. 此代码所属的题目是否处于正在进行的比赛。如果此代码所属题目处于正在进行中的比赛，则需要相应更新比赛相关数据表。

4. 此代码是否已经被评测过。由于数据更正等原因，可能需要重新评测某份代码；其后对数据库的更新流程也有所不同。

5.3.2 准备调用核心所需参数

根据评测核心在设计上的约定，封装层需要通过命令行的方式提供多个参数，包括语言类型、源文件路径、题号、数据目录、临时目录、时间限制、内存限制、输出限制等。每个参数的实际含义和单位等信息参见评测核心的说明文档。

5.3.3 调用评测核心进行评测

此阶段使用 5.3.2 节准备好的参数，调用评测核心进行评测。

5.3.4 获取并解析评测核心的输出

根据评测核心在设计上的约定，以其返回值和输出来标识一次评测的结果。如果核心的返回值非 0，表示在评测过程中出现错误，属于系统错误；否则可以从核心的输出字符串中读取出运行结果、内存占用、时间占用等信息。

5.3.5 更新数据库

此阶段根据数据库的设计和运行时的状态对数据库进行更新。

在更新数据库时需要考虑本次评测提交用户身份（普通用户、管理员）、所属题目是否处于比赛、是否是重新评测以及本次的评测结果对数据库的影响来抉择，涉及到的情况比较复杂，限于篇幅不一一展开。

在具体的实现中，由于需要连续执行多条数据库操作语句，因此采用了事务方式进行，保证了数据的一致性。

5.4 评测核心封装层设计小结

由于本系统在 Web 端，尤其是数据库的设计上（为了性能优化）实现得较为复杂，封装层在设计和实现上的复杂度在本系统中也属于较高的部分；由于封装层需要完成是整个系统最核心的业务逻辑，因此其重要性也是不言而喻的。在此模块的设计上，尽量倾注了较多时间和精力，实现了一个完整可用的封装层，但是具体的代码较为丑陋，且尚未充分地对其进行测试。

第 6 章　监听守护进程的设计与实现

如 2.5 节所述，监听守护进程（Daemon）采用了一个事件驱动模型：每当用户提交代码时，Web 端主动通知 Daemon，Daemon 载入评测核心封装层，以触发一个完整的评测流程。这样的事件驱动模型使得 Daemon 的运行更加有效率，且能够及时相应请求。

6.1　背景知识

6.1.1　守护进程

守护进程（daemon）是指在 UNIX 或其他多任务操作系统中在后台执行的电脑程序，并不会接受电脑用户的直接操控。系统通常在启动时一同起动守护进程。守护进程为对网络请求，硬件活动等进行响应，或其他通过某些任务对其他应用程序的请求进行回应提供支持。守护进程也能够对硬件进行配置（如在某些 Linux 系统上的 devfsd），运行计划任务（例如 cron），以及运行其他任务。

6.1.2　服务器

服务器（Server）指的是一个管理资源并为用户提供服务的计算机软件，通常分为文件服务器（能使用户在其它计算机访问文件），数据库服务器和应用程序服务器。服务器采用系统提供的方式（通常是监听一个 TCP/UDP 端口）等待用户提交的请求并予以处理。

6.1.3　TCP

传输控制协议（Transmission Control Protocol, TCP）是一种面向连接（连接导向）的、可靠的、基于字节流的运输层（Transport layer）通信协议，由 IETF 的 RFC 793 说明（specified）。在简化的计算机网络 OSI 模型中，它完成第四层传输层所指定的功能，UDP 是同一层内另一个重要的传输协议。

6.1.4 线程池

线程池是一系列等待接受任务的线程的集合。在某些应用场景下，常常有大量需要处理的任务，每个任务交予一个线程处理。由于线程的创建和销毁需要耗费较多时间，因此，可以预先创建好一定数量的线程并使其处于等待状态，一旦有新任务抵达，则可唤醒其中一个或一些线程对其处理，可提高处理效率。

6.2 监听守护进程的设计

由于 TCP 协议提供了面向连接且可靠的通信服务，而服务器采用了事件驱动机制，因此，完成一个 TCP 服务器来实现 Daemon，是一个最简单且合适的方案。

由于 Java 提供的 Socket、线程等库都对系统底层提供的相应机制进行了比较完善的封装，同时在语言层次上提供了线程之间的同步和互斥机制，因此，本系统采用 Java 进行开发，能够使 Daemon 运行地更稳定。

Daemon 的大体组成以及与其他模块的关系如图 6.1 所示。

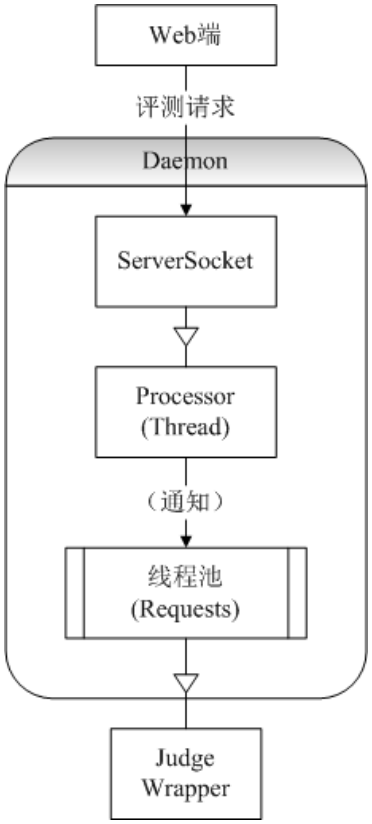

图 6.1　监听守护进程的结构及与其他模块的关系

在具体的设计和开发中，需要更仔细地考虑很多因素，包括但不限于：

1. TCP 服务器对请求的并发处理。在线评测系统在比赛的过程中通常会有大量的提交，为了尽快响应 Web 端的请求，应当使用多线程的方式来处理收到的连接。

2. 充分利用服务器的硬件资源。由于服务的硬件配置较高，尤其是中央处理器（CPU）的通常是多核心的，因此在设计时就应当考虑如何充分利用其处理能力。本系统采用了线程池的方案，既可以保证充分利用硬件资源，又能够保证不过度使用硬件资源而导致效率低下。

3. 线程之间的同步和互斥（生产者和消费者模型）。无论是多线程处理 Web 端请求（多个生产者），还是线程池调用 Judge Wrapper 完成评测（多个消费者），都需要特别注意线程之间的同步和互斥，保证 Daemon 的正常运行。

6.3 监听守护进程的实现

本系统设计了四个主要的类用于实现 Daemon 端的功能，分别是 Config、Daemon、Processor、Request。下面将分别阐述这几个类的设计与实现。

6.3.1 配置类（Config）

Config 类主要有两个功能：

1. 载入文件中的配置。由于守护进程所处的环境随时可能变化，因此将需要变化的配置独立到配置文件中，使得当环境变化时只需要修改配置文件即可，不需重新编译，可定制化更强。文件中主要包括了端口号、线程池的大小、JudgeWrapper 的路径和日志文件路径等参数。

2. 提供运行时的配置。守护进程运行时所需的参数直接从 Config 类中获得。

6.3.2 守护类（Daemon）

Daemon 类是提供了守护进程的入口和主干。在 Daemon 类的静态方法 main 中完成了配置的载入、线程池（Request 对象）的初始化等操作，并且创建了一个 SocketServer 监听服务器的某个端口，等待 Web 端的请求。当收到 Web 端发送的请求时，将创建一个 Processor 对象处理该请求。

6.3.3 请求处理类（Processor）

Processor 类继承于 Thread 类，工作是解析 Web 端发送的请求，读取请求中的代码编号，将其通过 Request 类提供的方法加入维护的待处理代码编号队列中。

6.3.4 线程池类（Request）

Request 类维护了一个带处理代码编号队列。所有对该队列的处理都必须通过其提供的 Synchronized 方法来完成，保证了生产者（Processor 对象）和消费者（Request 对象）之间的正常秩序。该方法在检测到待处理代码队列不为空时，会激活一个 Request 对象（即线程池中的一个线程）去处理请求。

由于 Request 类继承于 Thread 类，因此每个 Request 对象都是一个线程。在设计上，守护进程启动时即创建多个（数量在配置文件中设定）Request 对象组成线程池。每当有新的代码需要评测时，就会有 Request 对象被唤醒，载入 Judge Wrapper 进行评测；当评测结束后，该线程进入线程池挂起，等待下一次任务。

6.4 监听守护进程的安全问题

作为一个 TCP 服务器，由于需要处理来自外部的数据，应当尤其注意其安全。本系统目前采用了比较基本的方式来对请求进行验证：采用 IP 认证的方式来检验请求的合法性。如果发现请求者的 IP 地址不在可信列表内，则拒绝该请求；对请求的数据格式进行检查，如果发现格式不符合要求，即拒绝该请求。

6.5 监听守护进程设计小结

由于采用和合理的设计方案来实现监听守护进程，实际的开发非常顺利，在少量测试后即运行平稳，至今没有出现过问题。

但是，目前的设计方案也存在一些不足。某些不可抗因素，如 Web 端与 Daemon 之间通信失败、过量请求阻塞等会导致部分请求被遗漏，而 Daemon 本身并不操作数据库，无法获知被遗漏的请求。为了让系统更健壮地运行，实际上应当再增加一个附属模块，定时检查此前一段时间内被遗漏的请求并对其进行处理。

结论

本课题实现的在线评测系统目前已完成，并运行在武汉大学 ACM/ICPC 集训队的公开服务器上，为所有 ACM/ICPC 参与者与其他程序设计爱好者提供服务。

本文的主旨是详细地阐述一个完整的在线评测系统设计与实现的过程，系统、全面地分析目前设计的优劣、存在的问题，尤其是安全问题及其解决方案。通过本文详细论述的设计实现后，最终实现版本在面对全面的单元测试和大量的用户盲测，稳定高效的通过了各种正确、不正确及攻击性用户程序的测试，并且保证了系统安全，达到了预期的目标。

本系统开发人员通过将整个在线评测系统分割成若干松耦合的服务模块，模块间通过适当的接口进行通信，充分借鉴了前人总结的设计模式等经验，对设计完整正确实现后，达到了软件工程中追求的高内聚、低耦合的目标，降低了系统潜在问题的可能性，并让未来的维护人员能清晰明了的理解系统结构和逻辑流程，降低维护成本，且对未来的继续开发提供了可扩展的基础。

受开发时间的限制，本系统各个模块依然存在较大的改善、扩充的空间，如在 Web 端加入更多功能、加入缓存优化；在评测核心中加入更多的语言支持，主要是解释执行类语言，对系统调用过滤表的进一步细化和明确；进一步完善开发文档，为以后的维护和继续开发提供更便捷的方式等等。

对本文提出的目前各在线评测系统单服务器模式在负载高峰时服务压力过大导致无法及时响应的问题，本系统在设计上也预留了充分的扩展空间，使得将整个服务拆分为服务集群模式的设计简单可行。

总之，在线评测系统是为程序设计竞赛及教学服务的一个辅助平台，无论本系统如何改进，课题灵魂思想还是希望能尽可能的服务用户，让用户提高其程序设计能力，从而在相关领域获取更大的竞争力。本系统将延续武汉大学当前的在线评测系统在对武汉大学相关教学及 ACM/ICPC 竞赛训练中发挥的巨大作用，并力争以后能更好的发挥作用。

参考文献

[1] (美)W.Richard Stevens, Stephen A. Rago 著, 尤晋元、张亚英、戚正伟译，UNIX 环境高级编程（第二版）[M], 机械工业出版社, 2006

[2] (美) Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 著，李英军、马晓星、蔡敏、刘建中等译，设计模式：可复用面向对象软件的基础[M], 机械工业出版社，2000

[3] (美)Samuel P.Harison III, Guy L. Steele Jr. 著，邱仲潘等译，C 语言参考手册[M]，机械工业出版社，2003

[4] 王欢、杨树青著，Linux 环境下 C 编程指南[M]，清华大学出版社，2007

[5] 王腾、姚丹霖，Online Judge 系统的设计开发[J]，计算机应用与软件，2006(12)

[6] 周高嶔，基于白箱测试的源代码在线评测系统[D]，北京化工大学，2005

[7] 苑文会，基于黑箱测试的源代码在线评测[D]，北京化工大学，2005

[8] 李文新、郭炜，北京大学程序在线评测系统及其应用[J]，吉林大学学报(信息科学版)， 2005(S2)

[9] 唐俊武、南理勇、左强，在线考试系统开发中的几个问题及解决方法[J]，计算机与数字工程，2005(08)

[10] 毛华、罗朝盛，基于 Web 的在线考试系统的设计与实现[J]，计算机时代，2005(11)

[11] Pradeep Padala, Playing with ptrace, Part I[OL], Linux Journal, 2002(11)，http://www.linuxjournal.com/article/6100, 2010.4

[12] Pradeep Padala, Playing with ptrace, Part II[OL], Linux Journal, 2002(12), http://www.linuxjournal.com/article/6210, 2010.4

致谢

在本课题的开发过程历时逾一个半月，其间得到了许多同学老师的帮助，使得本课题的开发能够一步一个脚印踏实地走完。

感谢担任我毕业设计的指导老师董文永老师。董老师身为武汉大学 ACM/ICPC 集训队总教练，自我加入 ACM/ICPC 集训队起，就给予我非常多的帮助和支持。在本课题的开发过程中，更是常常关心项目进度，给出大量指导意见，推动了项目的开发。

感谢 Noah、Oak、Flood、HustOJ 等在线评测系统的开发设计人员杨传辉、吴永坚、刘高杰、杨宝奎、董超、高双星、叶文、马陈吉尔、刘潜、王良晶等师兄。是他们的工作激发了我对在线评测系统的兴趣，学习了必需的相关知识。与他们的交流使得我能够更深入地学习、理解这些系统的设计方法，总结设计经验，才能够提出我自己的设计思路。尤其是叶文和王良晶师兄，在本课题的开发过程中给出了非常重要的指导和建议，使得本项目的开发能够更加顺利地进行。

感谢刘阳同学对此项目做出的贡献。他帮助完成了评测核心中针对 Java 语言的关键部分，同时大量 Review 了其余部分代码，指出了其中存在的错误，并进行部分测试，直接推动了本项目的进展。

感谢武汉大学 ACM/ICPC 集训队，提供了开发和运行此课题的环境，以及所有帮助测试过此系统的集训队员和其他同学，感谢他们提出的大量建议和反馈，使得本项目更加完善、易用。

感谢陈璐希、黄轩、牛献会、杨欣郁、朱承兴、杜欢以及其他众多我无法一一枚举的同学和朋友在我完成毕业设计过程中对我的关心、帮助和支持。感谢我的家人、老师，感谢计算机学院、武汉大学，让我有机会、有条件在这样的环境中学习、成长。

感谢上述所有人，以及其他所有帮助过我的人，祝你们身体健康，学习工作一切顺利。

冯敏

二〇一〇年五月

于武汉大学

附录

附录一　数据库设计

```
DROP DATABASE IF EXISTS `land`;
CREATE DATABASE `land`;
USE `land`;

CREATE TABLE `problems`
(
  `problem_id` int primary key auto_increment,
  `title` varchar(200) NOT NULL,
  `description` text NOT NULL,
  `input` text,
  `output` text,
  `sample_input` text,
  `sample_output` text,
  `hint` text,
  `source` text,
  `contest_id` int NOT NULL default 0,
  `contest_seq` int NOT NULL default 0,
  `time_limit` int NOT NULL default 1000,
  `memory_limit` int NOT NULL default 65536,
  `spj` tinyint NOT NULL default 0,
  `accepted` int NOT NULL default 0,
  `submitted` int NOT NULL default 0,
  `enabled` tinyint NOT NULL default 1,
  `reserved` text
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1001;
```

```
INSERT INTO `problems`
(`problem_id`, `title`, `description`, `input`, `output`, `sample_input`, `sample_output`)
VALUES
(NULL, "A+B Problem", "Calculate a + b", "Two integer a, b (0<=a, b<=10)", "Output
a + b", "1 2", "3");

CREATE TABLE `users`
(
  `user_id` int primary key auto_increment,
  `username` char(20) NOT NULL,
  `password` char(50) NOT NULL,
  `nickname` varchar(100),
  `email` varchar(100),
  `school` varchar(100),
  `reg_time` datetime,
  `last_ip` char(20) default NULL,
  `submit` int NOT NULL default 0,
  `solved` int NOT NULL default 0,
  `enabled` tinyint NOT NULL DEFAULT 1,
  `share_code` tinyint NOT NULL DEFAULT 1,
  `group_ids` varchar(200) DEFAULT "",
  `solved_list` text
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
ALTER TABLE    `users` ADD INDEX (    `username` );

INSERT INTO `users`
(`user_id`, `username`, `password`, `nickname`, `email`, `school`, `reg_time`,
`group_ids`) VALUES
(NULL, "root", MD5("123456"), "root", "acm@whu.edu.cn", "whu", "2010-3-22
```

```
14:29:00", "1"),
(NULL, "anonymous", MD5(""), "anonymous", "acm@whu.edu.cn", "whu",
"2010-3-22 14:29:00", "");

CREATE TABLE `groups`
(
  `group_id` int primary key auto_increment,
  `group_name` char(50) NOT NULL,
  `description` varchar(200),
  `view_src` tinyint NOT NULL DEFAULT 0,
  `private_contest` tinyint NOT NULL DEFAULT 0,
  `admin` tinyint NOT NULL DEFAULT 0,
  `reserved` text NOT NULL default ""
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

INSERT INTO `groups` VALUES
(1, "root", "privileged user group", 1, 1, 1, "");

CREATE TABLE `sources`
(
  `source_id` int auto_increment,
  `problem_id` int NOT NULL,
  `user_id` int NOT NULL,
  `username` char(20) NOT NULL,
  `contest_id` int NOT NULL DEFAULT 0,
  `source_code` text,
  `length` int NOT NULL DEFAULT 0,
  `submit_time` datetime,
  `submit_ip` char(20),
  `lang` tinyint NOT NULL,
```

```
  `share` tinyint NOT NULL,
  `judge_time` datetime,
  `memory_usage` int NOT NULL,
  `time_usage` int NOT NULL,
  `result` int NOT NULL DEFAULT 0,
  `extra_info` text,
  primary key(`source_id`, `problem_id`, `user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE `admin_sources`
(
  `source_id` int auto_increment,
  `problem_id` int NOT NULL,
  `user_id` int NOT NULL,
  `username` char(20) NOT NULL,
  `contest_id` int NOT NULL DEFAULT 0,
  `source_code` text,
  `length` int NOT NULL DEFAULT 0,
  `submit_time` datetime,
  `submit_ip` char(20),
  `lang` tinyint NOT NULL,
  `share` tinyint NOT NULL,
  `judge_time` datetime,
  `memory_usage` int NOT NULL DEFAULT 0,
  `time_usage` int NOT NULL DEFAULT 0,
  `result` int NOT NULL DEFAULT 0,
  `extra_info` text,
  primary key(`source_id`, `problem_id`, `user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `contests`
(
  `contest_id` int primary key auto_increment,
  `title` varchar(200),
  `description` text,
  `private` tinyint NOT NULL DEFAULT 0,
  `start_time` datetime,
  `end_time` datetime,
  `enabled` tinyint NOT NULL DEFAULT 1
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE `user_at_contest`
(
  `user_id` int NOT NULL,
  `username` char(20) NOT NULL,
  `contest_id` int NOT NULL,
  `accepts` int NOT NULL DEFAULT 0,
  `penalty` int NOT NULL DEFAULT 0,
  `info_json` text NOT NULL,
  primary key (`user_id`, `contest_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE `problem_at_contest`
(
  `problem_id` int NOT NULL,
  `contest_id` int NOT NULL,
  `contest_seq` int NOT NULL DEFAULT 0,
  `AC` int NOT NULL DEFAULT 0,
  `PE` int NOT NULL DEFAULT 0,
  `CE` int NOT NULL DEFAULT 0,
```

```
  `WA` int NOT NULL DEFAULT 0,
  `RE` int NOT NULL DEFAULT 0,
  `TLE` int NOT NULL DEFAULT 0,
  `MLE` int NOT NULL DEFAULT 0,
  `OLE` int NOT NULL DEFAULT 0,
  `total` int NOT NULL DEFAULT 0,
  primary key(`problem_id`, `contest_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE `mails`
(
  `mail_id` int primary key auto_increment,
  `from_user_id` int NOT NULL DEFAULT 0,
  `from_username` char(20) NOT NULL,
  `to_user_id` int NOT NULL DEFAULT 0,
  `to_username` char(20) NOT NULL DEFAULT 0,
  `send_time` datetime NOT NULL,
  `title` char(100),
  `content` text,
  `unread` tinyint NOT NULL DEFAULT 1,
  `reader_del` tinyint NOT NULL DEFAULT 0,
  `writer_del` tinyint NOT NULL DEFAULT 0
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

附录二　项目代码

见附件。项目托管于 Google Code，也可到 http://woj-land.googlecode.com 下载。