**深 圳 大 学**

本 科 毕 业 论 文（设计）

题目:**深圳大学在线程序实时测评系统的设计、搭建和开发**

**——网站后台和测评系统后台服务的设计与实现**

姓名: **张佳楠**

专业: **计算机科学与技术**

学院: **计算机与软件学院**

学号: **2011150049**

指导教师: **张小健**

职称: **讲师**

2015年 04月 19日

**深圳大学本科毕业论文（设计）诚信声明**

本人郑重声明：所呈交的毕业论文（设计），题目《深圳大学在线程序实时测评系统的设计、搭建和开发——网站后台和测评系统后台服务的设计与实现》 是本人在指导教师的指导下，独立进行研究工作所取得的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明。除此之外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。本人完全意识到本声明的法律结果。

毕业论文（设计）作者签名：

日期： 年 月 日

### 目录

[摘要 1](#_Toc417494806)

[1前言 1](#_Toc417494808)

[1.1 ACM/ICPC 简介 1](#_Toc417494809)

[1.2程序在线实时测评系统简介 1](#_Toc417494810)

[1.3 程序在线实时测评系统用途及意义 1](#_Toc417494811)

[1.4 本系统开发搭建的意义 2](#_Toc417494812)

[1.5 本文主要的研究内容 2](#_Toc417494813)

[2 开发程序在线实时测评系统技术和搭建环境 2](#_Toc417494814)

[2.1 基于Python 的Web框架Django 2](#_Toc417494815)

[2.2 Linux 系统相关编程 3](#_Toc417494816)

[2.3 搭建环境和工具选择 3](#_Toc417494817)

[3 Online Judge的需求分析和总设计 3](#_Toc417494818)

[3.1 主要目标及合作分工 3](#_Toc417494819)

[3.2 Web Platform后台需求分析 4](#_Toc417494820)

[3.2.1 Web Platform 后台的主要功能 4](#_Toc417494821)

[3.2.2 Web Platform 后台主要解决的问题 4](#_Toc417494822)

[3.2.3 Web Platform 后台开发运行环境 5](#_Toc417494823)

[3.3 Judge System后台服务需求分析 5](#_Toc417494824)

[3.3.1 Judge System 后台服务的主要功能 5](#_Toc417494825)

[3.2.2 Judge System 后台服务主要解决的问题 5](#_Toc417494826)

[3.2.3 Judge System 后台服务开发运行环境 6](#_Toc417494827)

[3.3 Online Judge 总设计 6](#_Toc417494828)

[4 Web Platform 的设计和实现 8](#_Toc417494829)

[4.1 Web Platform 总设计 8](#_Toc417494830)

[4.1 数据库设计 8](#_Toc417494831)

[4.2 功能模块 8](#_Toc417494832)

[4.2.1 用户模块 8](#_Toc417494833)

[4.2.2 训练模块 9](#_Toc417494834)

[4.2.3 比赛模块 10](#_Toc417494835)

[4.2.4 后台管理模块 10](#_Toc417494836)

[4.3 安全和权限 10](#_Toc417494837)

[5 Judge System 的设计和实现 11](#_Toc417494838)

[5.1 Judge System 总设计 11](#_Toc417494839)

[5.2 内核空间相关程序的详细设计及实现 11](#_Toc417494840)

[5.2.1 访问控制 11](#_Toc417494841)

[5.2.2 Sandbox机制 11](#_Toc417494842)

[5.2.3 IPC相关控制 13](#_Toc417494843)

[5.2.4 文件系统相关控制 13](#_Toc417494844)

[5.2.5网络相关控制 13](#_Toc417494845)

[5.2.6记录系统资源 13](#_Toc417494846)

[5.3 用户空间相关程序的详细设计及实现 14](#_Toc417494847)

[5.3.1 Judge任务监控 14](#_Toc417494848)

[5.3.2 处理用户程序 14](#_Toc417494849)

[5.3.3 运行测试用户程序 15](#_Toc417494850)

[6 总结 15](#_Toc417494851)

[致谢 17](#_Toc417494852)

[Abstract 18](#_Toc417494853)

**深圳大学在线程序实时测评系统的设计、搭建和开发**

——网站后台和测评系统后台服务的设计与实现

计算机科学与技术 张佳楠

学号：2011150049

【摘要】本文是在线程序实时测评系统后台服务部分的设计与实现，包括网站后台和测评系统后台。在Linux操作系统环境下，使用基于python的Django框架实现网站的高并发、高安全访问，应用系统提供的C语言函数和内核系统调用，创建一个高安全性、高稳定性、高精确性的测评系统后台服务端。论文主要是从需求分析与系统设计等方面对测评系统的功能设计进行了说明，再从网站后台功能和测评系统后台服务端功能这两方面展示在线程序实时测评系统的部分实现具体细节。

【关键词】Web 后端；Django；在线测评系统；Linux内核系统调用；ACM/ICPC

# 前言

## ACM/ICPC 简介

ACM国际大学生程序设计竞赛（International Collegiate Programming Contest）ACM/ICPC， 是由国际计算机界历史最悠久、最具权威性的组织ACM学会（Association for Computer Machinery）主办，是世界上公认的规模最大、水平最高的国际大学生程序设计竞赛，其目的旨在让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。

## 1.2程序在线实时测评系统简介

程序在线实时测评系统（Online Judge）的核心，其主要功能是对用户在Web网页提交的程序源代码进行编译和执行，并通过预先设计的多组测试数据来检验提交程序的正确性，所需的内存空间，以及需要使用的CPU运行时间，最后将结果返回给用户。

## 1.3 程序在线实时测评系统用途及意义

Online Judge 系统广泛应用于程序设计竞赛及训练中，其中以ACM国际大学生程序设计竞赛形式的比赛最为出名。

随着ACM/ICPC在全球的推广，国内不少大学都建立了程序在线实时测评系统著名的如有西班牙University de Valladolid 建立的UVa Online Judge、俄罗斯的Ural州立大学的Online Judge、2006年ACM/ICPC世界冠军SGU的Online Judge；国内的北京大学、浙江大学、天津大学等。这些测评系统对这些学校在ACM/ICPC的发展起了不可估量的作用。

## 1.4 本系统开发搭建的意义

深圳大学从2003年参加广东省程序设计竞赛（GDCPC）和ACM/ICPC亚洲赛区预赛，

取得了一些成绩，但是与一流强队之间还有很大的差距，为了缩短这种差距，建立一个良好的训练环境，进一步提高深圳大学ACM队的实力，同时扩充ACM/ICPC在学校范围内的影响，提高全校对计算机编程有兴趣的同学的运用计算机这一现代工具解决问题的的能力，建立一个先进的程序在线实时测评系统势在必行。在2006年，实验室师兄组队开发了一个程序在线实时测评系统，但是由于系统开发时没有考虑周全，存在难迁移和难维护，以及并发性和高可用不足等问题，经过这么多年，已无法满足实验室队员和同学们的需求。所以为了让ACM队员和学校同学能够积极的锻炼自己的编程能力，需要重新设计开发和搭建一个不一样的程序在线实时测评系统。

## 1.5 本文主要的研究内容

本文主要的研究内容是程序在线实时测评系统后台的设计，开发和搭建，包括Web 网站（Web Platform）和 判题系统（Judge System）。具体内容是如何开发和搭建高并发、高安全、高可用的程序实时在线测评网站，如何开发和搭建高安全性、高稳定性、高精确性的测评系统后台服务端。

# 开发程序在线实时测评系统技术和搭建环境

## 基于Python 的Web框架Django

Django 是一个高效，高层次，开源代码的Python Web 框架，鼓励开发人员进行快速开发和简单实用的设计。由于Django是由经验丰富的开发人员创建的，所以它特别专注简化在很多Web开发上的麻烦并进行接口封装复用，让Web 开发者能够更专注于编写清晰、易维护的应用程序的核心代码，而无需从头做起。

Django通过提供通用Web开发模式的高度抽象和频繁进行的编程作业的快速解决方法来使得开发复杂的、数据库驱动的网站变得简单。

Django采用 MVC （模型-视图-控制器）的软件开发方法，在Django框架中，Model 代表数据存取部分，由Django 数据库层处理，View代表选择哪些数据显示和怎样显示的部分，由Django 的视图View 和 模板Template处理，Controller 代表根据用户输入选择哪些视图部分，由Django框架根据统一资源定位器配置文件的设置，对给定的url 调用正确的视图函数。由于控制器是由Django 框架自动帮助开发者封装处理了，开发者在控制器的开发中只需简单的配置一下url的路由规则，因此遵循MVC模型开发的Django框架其实应该被称为一种MTV框架，其中的T代表Template（模板）,在Django中是对Web页面和其他类型文档进行渲染显示，在开发程序在线实时测评系统中，我们弃用Django 原生自带的模板，而采用自行扩展后的渲染速度比较快的Mako模板进行开发，并结合前端开发框架AngelarJS进行Web页面的显示。

## Linux 系统相关编程

由于Judge System 是搭建在Linux 下，用户程序是在Linux 下编译和运行的，所以本系统必须调用Linux 系统所提供的函数库来进行编程。

本系统主要调用包括以下方面的函数：

1. 文件IO和文件目录，主要是来对文件的读、写等操作和对文件或目录的权限设置，以及工作目录环境的切换等。
2. 进程线程，主要是来对用户编译运行的程序创建执行终止，以及资源控制等。
3. 数据库，主要是来查询数据库，查询用户是否有提交程序，已经记录用户程序的状态等。

## 搭建环境和工具选择

本程序在线实时测评系统完全搭建在Linux服务器上，搭建主要分成2个部分：数据库服务器搭建，Web 服务器搭建。在工具选择上，数据库MySQL，Http服务器Niginx，Django部署方式Uwsgi。由于本系统是由本人和另一个同学共同开发的，需要进行代码同步等工作，故我们选择了SVN做代码同步工具。

数据库MySQL是一个免费的关系型数据库管理系统，在行业内的普及度非常高，出现问题比较容易找到解决方案。MySQL比其他数据库更容易学习，更容易上手，在本系统中，数据库主要用于简单的存储记录，不需要过多的功能，与本系统使用的Web框架能够很好的配合。在搭建中，只需要基本按照和配置成只有Web服务器和Judge System 服务器可访问。

Niginx是由俄罗斯人开发的一种实现web服务器的工具，与常用的Apache优势主要是支持高并发，低内存占用。Uwsgi是一个用C语言开发的wsgi服务器，旨在提供专业的Python web 应用发布和开发，主要特点是超快性能，低内存占用，多App管理，详细日志和高度可定制。本系统是基于Nginx 和uWsgi 来部署Django项目，Nginx 接收Http请求，静态文件和静态页面请求Nginx直接处理，动态页面请求交给uWsgi处理，uWsgi在配合Django路由进行处理。

SVN（Subversion）是一个软件开发中用于版本控制的系统，采用分支管理系统，其优势在于每次提交都是原子操作，无论一次提交是单个文件还是多个文件，都是作为一个整体提交的，重命名、复制、删除文件等动作都会保存在版本历史记录当中，分支的开销非常小，而且简单方便易用。在本系统开发中，很多时候都会有测试demo，利用svn的分支与还原历史版本可以更容易地进行测试和防止代码混乱。

# 3 Online Judge的需求分析和总设计

## 3.1 主要目标及合作分工

本Online Judge包括后台判题系统和Web网站平台两大版块，由本人和另一位同学共同完成。以下为对该两大版块的总介绍：

1. 后台编译系统：仅负责对源代码的编译、运行、判定，源代码从公有数据库处查询提交记录，并且在判定完毕后把结果写回到公有数据库。
2. Web网站平台：负责对系统各类信息的综合管理，包括整个OJ网站的展示及与后台编译系统的沟通

说明：后台编译系统和Web网站平台是通过数据库进行耦合的，以上公有数据库是指后台编译系统和Web网站平台的共同数据库  
其中，本人负责后台编译判题系统的设计和开发以及Web后台部分的开发，主要目标如下：  
基于Linux系统编程、Django开发等后台手段来对用户程序在线测评功能予以实现，同时注重系统的Web后台和判题后台安全问题，降低系统风险。最后协同前端编写出一个能够进行在线程序评测的网站。

## Web Platform后台需求分析

### Web Platform 后台的主要功能

1. 提供邮件推送，用户注册时后台发送邮件到用户邮件进行验证；用户找回密码，通过后台发送邮件到用户注册邮箱进行密码重置；当管理员需要对用户进行通知时，可以通过邮箱推送信息。
2. 实时计算比赛排行榜和比赛问题提交记录的统计，让用户能够根据比赛排行和问题提交记录的统计选择性的做题。
3. 权限控制，适配不同用户有不同权限和不同展示页面，训练中隐藏问题只有管理员能够看到，训练中提交状态只能看到训练的，不能查看比赛的，普通用户只能查看自己提交的代码，管理员用户可以看到所有提交记录和所有用户提交的代码，比赛只能有比赛权限的用户可以访问，管理网站界面只能由管理员访问等等。
4. 提供文件系统，网站管理中管理员能够实时的对问题的测试数据进行添加、删除和修改以及查看，即需要上传、下载和删除后台问题的测试数据文件。
5. 提供日志系统，方便开发者在以后的维护中监控Web Platform的异常，及时修补漏洞，以及方便开发者进行正式环境上的测试和调试。
6. 保证Web Platform 的高并发和安全性。

### Web Platform 后台主要解决的问题

1. 如何简单方便的利用Django内置封装python库的邮件发送服务接口。
2. 采用何种方式将存储日志文件与Django日志配置结合。
3. 保证Web Platform后台服务端的安全，使得不会受到恶意提交程序和web 恶意攻击（主要是跨站脚本攻击XSS和跨站请求伪造攻击CSRF）的破坏，从而导致网站崩溃，或者测试数据文件，用户信息，用户源代码等重要数据被盗取。
4. 如何快速计算比赛排行榜和比赛问题提交记录的统计而不影响用户查看排行榜和提交记录统计的体验。
5. 允许不同用户拥有不同视图，控制用户权限，细分用户权限等级，防止越权查看其他用户信息和其他用户提交的源代码。
6. 能够支持大测试数据文件的上传和下载，并且防止因为文件过大而占用服务器过多内存，使得服务器崩溃。

### 3.2.3 Web Platform 后台开发运行环境

Web Platform后台采用的是Linux操作系统，具体是Centos 7.0，内核版本3.10.0，基于python语言的Web 框架Django 进行开发。

## 3.3 Judge System后台服务需求分析

### 3.3.1 Judge System 后台服务的主要功能

1. 监控数据库在Web Platform 上的提交记录。
2. 实时同步Web Platform 上每个问题的多组测试数据文件，包括测试数据文件的增加，删除和修改。
3. 对用户在Web Platform 上要求进行测试的程序，从数据库中提取出该程序的源代码，对源代码进行编译，并将编译信息（编译出错信息）保存到数据库中。
4. 从数据文件目录中提取出对应的测试数据，运行通过编译后的程序，利用测试数据对其进行黑盒测试，并同时控制和记录对系统资源的使用情况，包括使用的内存空间，CPU运行时间，堆栈的大小，程序终止的原因等。
5. 从数据库查询是否是特殊判断，若是则提取相对应的判定程序（Special Judge Program），利用该判定程序，否则直接利用比较程序，对用户程序的运行输出进行分析，判定其是否能通过测试，并将最终判断结果保存到数据中。
6. 保证系统的稳定性和安全性。

### 3.2.2 Judge System 后台服务主要解决的问题

1. 采用何种方式进行数据库监控，保证后台和前台的数据可靠性和一致性。
2. 采用何种方式将Judged System 后台和Web Platform上每个问题的测试数据文件进行同步。
3. 保证程序测评系统后台服务端的安全，使得不会受到恶意提交程序的破坏，从而导致系统瘫痪，或者测试数据文件等重要数据被盗取。
4. 比较准确的控制和记录程序运行时所使用的系统资源。当使用系统资源超过限制后，程序将被强行中止。否则，记录其在运行过程中，最多的时候，使用了多少系统资源。（这里控制的系统资源主要包括输出文件大小，堆栈空间大小，内存空间，CPU时间，子进程和线程数等， 记录的系统资源主要指的是内存空间及CPU处理时间）
5. 允许多个提交的程序同时运行，即并行判题，提高服务器的利用率，减少用户等待时间。

### 3.2.3 Judge System 后台服务开发运行环境

Judge System 后台服务采用的是Linux操作系统，具体是Centos 7.0，内核版本3.10.0，用C语言开发，也包含部分bash脚本。

## Online Judge 总设计

本程序在线实时测评系统在系统架构上总有三大模块：Web Platform（程序提交网站平台）, Data Base（数据库） 和 Judge System（程序测评系统）。Web Platform 主要是负责客户端的Web 请求，Judge System 后台服务主要是负责编译用户程序，运行用户程序以及评测用户程序的正确性。这两个模块通过访问共同的Data base来交互。Web Platform 设计是基于最经典的设计模式B/S（Browser/Server，浏览器/服务器模式）来进行Web 服务器设计。Judge System 后台服务是基于Linux 系统编程而通过创建多线程，多进程来并发处理程序。整个系统采用类似于分布式系统思想进行设计，在保证系统的安全性，稳定性，高并发的同时也保证了能够最大程度进行扩展，如可以在用户提交过多程序时增加更多判题服务器来及时响应用户的测试程序。在设计上充分考虑了系统的可靠性，安全性，容错性，可扩展性，实用性以及易操作性，主要体现在系统能够安全可靠地长时间运行，能够有一定程度抗攻击，能够有较强的容错能力，能够在兼容原系统的基础上软件和硬件有一定程度上地扩展能力，能够有良好的用户体验。

本程序在线实时测评系统总系统架构图如下：

Web Platform

（Web服务端）

用户（浏览器）

Http 请求

Http请求响应

查询DB 写入DB

写入评测结果

数据库

Judge System

（判题系统）

查询提交记录

图1 系统架构图

整个系统在线实时评测流程如下：

1. 用户在web 网页上选择题目，本地编写代码并通过网页进行提交。
2. Web Platform 后端响应用户提交代码，并写入数据库中。
3. Judge System 开启监控数据库用户提交记录，当监测到有用户提交记录时，将创建judge program 子进程来处理用户提交程序。
4. Judge Program 进程查询数据库相关信息，包括题目时间内存限制，用户信息，提交语言等，设置相关系统资源的限制和系统函数的调用限制，进行编译程序。
5. 若编译错误，直接将编译错误信息和评测结果回写到数据库。
6. 编译正常，将开始运行程序，并通过对比程序或特殊判断程序来评测用户程序的正确性，并将最终结果回写到数据库中。
7. 用户通过刷新页面来向Web Platform请求评测结果， Web Platform 通过查询数据库获取评测结果并返回结果来相应用户的请求。

整个系统在线实时评测流程图如下：

用户提交程序

生成源程序文件

编译用户程序

编译是否通过

否

是

运行用户程序

运行是否正常

否

是

比较运行结果

运行错误

编译错误

结果保存数据库

图2 测评系统流程图

# 4 Web Platform 的设计和实现

## 4.1 Web Platform 总设计

Web Platform 后台部分主要可以分成4个部分：

第一部分，是用户模块处理，其主要功能有：用户登录验证和存储登陆态，用户注册和注册验证邮箱，用户忘记密码通过邮箱重置密码。

第二部分，是训练模块处理，其主要功能有：训练问题列表筛选处理，用户训练排名，用户提交记录以及代码提交。

第三部分，是比赛模块处理，主要功能有：比赛问题列表展示，比赛实时排行榜的计算以及比赛实时问题统计。

第四部分，是后台管理模块处理，主要功能有：添加问题，添加比赛。

## 4.1 数据库设计

本系统中数据库是Web Platform 和Judge System 之间通讯的介质，所以数据库的设计非常重要。本系统中数据库中最重要的三张表是用户表，问题表和提交记录表。一下是这三张表的结构的设计（数据库总共有24张表，这3张表是最为重要的）：

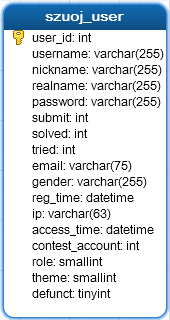
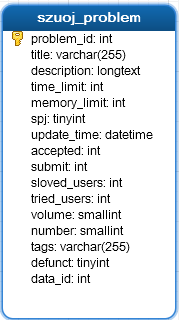
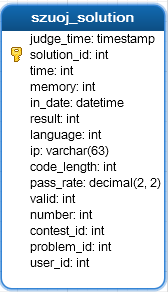
  

图3 用户表 图4 问题表 图5 提交记录表

## 4.2 功能模块

### 4.2.1 用户模块

在本系统中用户需要注册，登陆才能进行提交测试程序。

1. 首先是用户注册，在web后台当出现用户请求注册时，需要通过正则表达式和数据库查询验证用户提交申请注册的信息（用户名，密码，邮件）是否符合规定。当信息校验通过后，通过后台邮箱SMTP服务发送邮件（用户名和验证的SHA1码）到用户注册的邮箱中。用户收到邮件后需点击邮箱里的链接来进行验证。后台收到用户注册验证请求时，查询数据库是否是合法验证，是否能通过验证，验证通过后将用户等注册信息写入用户表和Django当中去。

其中的邮件SMTP服务是利用Django框架封装python原生库Email的接口，只需要在后台简单配置好服务，并将配置信息写在Django配置文件Settings中。

1. 用户注册并验证完后，用户可以进行登陆。登陆模块中我们采用用户名和密码匹配方式进行登陆，其中密码是经过Md5哈希密码加随机Salt加规定字符再Md5哈希来验证的。同时也利用了Django提供的方便对用户登陆验证的Auth/Auth验证系统。应用该验证系统主要是能够方便解决用户登录后的Session和Cookie问题。当验证通过后，Django的Auth/Auth验证系统会在后台存储userid的session ，并在每次用户请求中，先自行通过验证后再将user信息添加到request 请求中去。这样我们可以在request中获取user来判断是否用户登陆，以及登陆后用户的信息。这样使用Djaong的Auth/Auth验证系统可以不需要考虑session，cookie的失效等问题，只需要在Django的Settings配置失效时间等即可。
2. 如果用户忘记密码可以通过邮箱重置密码，但用户请求邮箱重置密码时，首先从数据库查询记录验证是否是用户名和邮箱对应，若对应，随机生成8位数的密码通过后台邮箱服务发送到用户注册邮箱中。这时用户即可用新的密码进行登陆，并通过用户信息修改页面进行密码的修改。

### 4.2.2 训练模块

在训练模块中，后台共实现了8个主要功能接口。

1. 查询问题列表名称接口。本系统中将所有问题分成一列一列，每一列最多有100个问题，这样可以方面用户查看和展示。该接口主要是查询数据库问题表有多少列问题列表，并通过转换将列表名以大写字母返回给前端。其中若是管理员，还能看到没有编号的问题Special Problem列（SP列）。
2. 查询问题列表接口，这个接口是跟上面接口对应的，当请求具体某一列时，该接口将查询数据库问题列表对应该列的问题，并通过对未登录用户，普通用户和管理员进行问题筛选再返回给前端。
3. 问题详细信息接口。该接口主要是查询问题的详细信息，包括标题，描述，Judge信息，以及用于统计饼图展示的提交情况信息。
4. 每个问题用户的排名接口。这个接口主要是查询每个用户对某问题的解决代码的内存，时间排名。
5. 用户训练排名接口。该接口通过查询每个用户总尝试问题和总解决问题的数量，将用户进行排名，这样能够跟清晰的知道每个用户的做题情况。
6. 测试程序提交接口。该接口是用来提交用户测试程序的。
7. 用户提交记录查询接口。该接口是用户查看用户提交的测试程序的正确与否。通过查询提交记录表，对未登录用户，普通用户和管理员进行记录筛选再放回给前端。
8. 查看提交记录详细信息接口。当用户提交测试程序后，可以通过该接口查看提交的源代码和编译错误信息（出现编译错误的情况下）。

### 4.2.3 比赛模块

在比赛模块中，很多接口都是跟训练模块公用的，所以后台共实现了4个比赛独立功能接口。

1. 查询权限接口。该接口仅仅是为了前端展示，判断是否有权限进入比赛，该接口与后台验证权限接口没有直接关系。
2. 查询比赛排行榜。该接口主要是查询比赛实时排行榜，通过数据库联合查询提交记录表，问题表和用户表，查询并计算出每个用户的每道题的提交情况，以及总的问题解决数和总的罚时时间。
3. 查询比赛问题统计表。该接口主要是查询比赛实时的问题统计，通过数据库联合查询提交记录表和问题表等，查询并计算每个问题的提交情况，包括通过次数，各种错误次数等。

### 4.2.4 后台管理模块

在后台管理模块中，总共有2个主要功能。

1. 添加和修改问题功能。该功能共分2部分，第一部分是添加问题和修改问题信息，通过管理员页面提交数据，后台创建或修改数据库记录，并将结果返回给管理员。第二部分是添加，删除和下载问题数据功能，由于该功能是涉及到大文件的上传和下载（测试数据问题可能非常大），在下载上本系统利用python语言提供的迭代器功能，让大文件可以分块的返回给客户端，这样的方法不会造成服务器内存占用过多而导致服务器崩溃。
2. 添加和修改比赛功能。该功能主要是创建或修改比赛，其中包括比赛人员的设置，比赛问题的设置以及比赛基本信息的设置。这个功能实现中，由于比赛人员，比赛问题，比赛信息都是设计在不同的表当中，需要解决同步问题。利用Django提供的数据库事务操作接口，当比赛人员，比赛问题，比赛信息写入数据库出现失败时，可以自动回退操作。

## 4.3 安全和权限

在安全方面，本web平台主要处理了3个安全模块问题。

1. Web访问数据库的安全，主要在SQL注入等问题，在这方面，本web后台不需要过多的处理。Django 采用ORM (对象关系映射)来操作数据库，其内置的ORM是由Django框架封装的操作数据库的API，使用Django 的ORM 只需要做简单的对象定义，无需直接操作数据库，无需学习SQL语句，无需了解数据库的运行机制等就能够自动生成数据库结构和操作数据库。在兼容性和扩展性上，Django 也提供了原始的SQL 语句执行的API，但是使用原始的SQL语句时需要开发者自行去SQL的各种安全问题（如SQL注入等），而是使用ORM技术，Django框架在数据层上已经自动处理了很多SQL的安全问题。由于ORM技术的使用可以保证在一定程度上的安全，在本测评系统中，我们也采用ORM的操作数据库的技术。
2. Web 页面上的XSS攻击，在这方面，本Web后台利用了Django框架提供的Miadleware（中间件）安全处理。中间件Miadleware 是对用户的每个Http 请求都执行相同的某段代码，而且执行的代码是在view处理之前。 本程序在线实时测评系统主要用中间件来处理html代码和javascript代码的xss注入过滤处理，以及特定url访问的过滤处理；其中那个html处理用到了python自带的HTMLParser转义库。同时XSS攻击在前端框架AngelarJS中也进行了一定程度上的防御。
3. Web页面请求CSRF攻击，在这方面，本Web后台利用了Django框架提供的CSRF中间件，很轻松就能够为系统提供CSRF防护能力。Django 提供的CSRF中间件是通过在服务端进行CSRF防御，在用户请求生成一个随机csrf\_token给前端，当用户提交请求时，对其进行验证csrf\_token的正确性。由于请求中带有csrf\_token，就无法通过跨域获取cookie进行伪造请求攻击。

# 5 Judge System 的设计和实现

## 5.1 Judge System 总设计

Judge System 后台服务，可以分为两大部分。

第一部分，是运行在内核空间的程序，其主要功能有：1）创建一个操作系统级别的简单沙箱，将进行测试的程序运行在该沙箱中，在该沙箱中运行的程序对其所有的系统调用（系统资源和系统函数等）都将受到管制；2）通过proc伪文件系统记录程序在运行过程中使用的最大系统资源，并在该程序的进程结束后返回记录的内容。

第二部分，是运行在用户空间中的程序，其中主要包括3个进程，1）监控数据库提交记录进程；2）预处理提交记录相关信息，运行用户程序并进行测试比较进程；3）监控问题测试数据同步进程，将问题测试数据文件进行实时同步，保证判题的正确性。各个进程之间利用信号量，共享内存，进行进程之间的同步以及通信。

## 5.2 内核空间相关程序的详细设计及实现

### 5.2.1 访问控制

通过对需求的分析，，得出本测评系统后台服务端，主要的安全薄弱环节是出在，允许在服务器上运行用户提交的程序。但是测评系统后台服务端无法避免的需要运行用户提交的程序，为了使本测评系统后台服务端能达到高安全性的要求，只能从操作系统入手寻找解决方案。

在Linux中所有的应用程序，都是通过调用Linux提供的系统调用（System Call）函数，来执行对设备、文件系统、网络、进程等的访问。如果我们在运行需要测试的用户测序时，能对Linux提供的系统调用函数都进行管制（仅限于有可能危害系统的函数），那么我们就可以消除系统由于运行用户提交的程序可能出现的各种各样安全问题。

### 5.2.2 Sandbox机制

本系统设计的沙箱主要有两个方面的功能：一是对系统调用的限制，二是对系统资源的限制。

1）限制系统调用设计和过程如下：

在UNIX中有上百个系统调用（进程控制，文件系统控制，系统控制，内存管理，网络管理，Socket控制，用户管理，进程通信），在访问模块上需要对其调用进行控制。况且其中很大一部分系统调用在运行用户提交的测试程序过程中不需要用到的，比如ls,， cp， mkdir，rm等，还有一小部分系统调用会对系统造成各种各样安全问题的，比如fork，kill，exec等，更甚至还有一些比如socket等网络调用会造成敏感信息，比如问题测试数据的泄漏等。 由于存在上述各种安全问题，所以需要限制在运行用户提交的测试程序过程中系统调用。

在本系统中，采用了 Linux提供了一种优雅的机制ptrace系统函数来实现在用户层上拦截和修改系统调用。Linux下的ptrace在这里是一个非常好用，非常强大的工具，它可以在用户态和内核态之间切换之前和之后，将进程暂停，以方便控制进程的处理，控制进程通过ptrace可以读取到当前进程想要去做什么，这样就可以在用户程序造成破坏之前将程序中止。

Ptrace提供了一种使父进程得以监视和控制其它进程的方式，它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。在测试程序进程中，我们在该父进程通过PTRACE\_TRACEME（父进程跟踪某个子进程的执行，任何传给子进程的信号将导致其停止执行，同时父进程调用 wait() 时会得到通告）作为ptraced的第一个参数进行来调用，进而跟踪用户程序运行进程，这样在执行系统调用之前，内核会先检查当前子进程是否处于被”跟踪“（traced）的状态，如果是的话，内核暂停当前进程并将控制权交给跟踪进程即父进程，使跟踪进程得以察看或者修改被跟踪进程的寄存器。接着通过PRACE\_GETREGS作为ptrace的第一个参数来调用，可以只需一次函数调用就取得所有的相关寄存器值（子进程执行的系统调用ID放在ORIG\_EAX），这时候只需要查看子进程执行的系统调用ID是否在预先设置的白名单之内，若不存在，则通过PTRACE\_KILL作为ptrace的第一个参数来调用，中止子进程。

其中限制非法系统调用一般有两种方法，一种是“通”，一种是“堵”，通方式其实就是设置白名单机制，堵方式就是黑名单机制，在本系统中使用白名单机制才能更好的进行安全控制，只允许用户提交的测试程序使用提前设置的白名单里的系统调用，对于其它系统调用，即使它是对系统无影响的，安全的，也不会被允许，比如ls，mkdir等。

特别说明：

操作系统提供了一种标准的服务来让程序员实现对底层硬件和服务的控制（比如文件系统），叫做系统调用(system calls)。当一个程序需要作系统调用的时候，它将相关参数放进系统调用相关的寄存器，然后调用软中断0x80，这个中断就像一个让程序得以接触到内核  
模式的窗口，程序将参数和系统调用号交给内核，内核来完成系统调用的执行。  
在i386体系中(本系统所有的代码都是面向i386体系)，系统调用号将放入%eax,它的参数则依次放入%ebx, %ecx, %edx, %esi 和 %edi。

2）限制系统资源的设计和过程如下：

在Linux系统中，进程资源通常是指CPU时间，内存空间，文件大小，子进程数量，堆栈空间等资源。在本系统中，我们利用Linux内核提供的setrlimt系统调用来限制进程所能得到的最大资源。在运行用户提交的测试程序前，通过setrlimt用参数RLIMIT\_CPU，RLIMIT\_FSIZE，RLIMIT\_NPROC，RLIMIT\_STACK，RLIMIT\_AS分别设置该进程的最大运行CPU时间，最大产生的文件大小，最大子进程数量，最大堆栈长度，最大内存空间。当运行测试程序进程超过设置的资源最大值时，则向父进程发送不同的信号（超内存，超时等信号），父进程通过wait4，WIFEXITED，WEXITSTATUS，WIFSIGNALED，WTERMSIG等系统调用接收子进程的发送的信号，若是超出设定资源的信号，则中止子进程。

### 5.2.3 IPC相关控制

IPC是指进程内部通信（Inter-Process Communication），是进程之间相互通信的其他技术。在沙箱控制下的进程，已经通过白名单方式禁止访问控制模块中响应控制函数，禁止其使用共享内存、信号量和消息队列。

### 5.2.4 文件系统相关控制

由于程序测评系统需要给予用户测试测序一定的文件读写权限，因此不能单纯的去禁止其读写文件的权限。换之，采用Linux系统提供的chroot()系统函数，使得用户测试程序的进程通过chroot机制来更改其所能看到的根目录，即将该进程限制在指定目录中，保证该进程只能访问该目录及其子目录的文件，在该目录下只有少数运行测试程序所需要的库文件和程序，并且所有文件的权限都设置妥当，只有root用户才有权限删除和修改以及改变文件性质。在改变了进程的根目录后，在利用setuid(), setgid()来改变改进程所属的用户以及用户所在的组别，使得其只有极少的文件写入权限，从而保证了整个测评系统服务器上其他文件的安全。

### 5.2.5网络相关控制

利用沙箱白名单方式禁止了在测试进程中做任何访问网络的操作，包括socket的创建（socket\_create）、绑定（socket\_bind）、监听（socket\_listen）、连接（socket\_unix\_stream\_connect），发送（socket\_unix\_may\_send）的网络相关操作。

同时在网络应用层方面，由于程序测评系统后台服务端不需要有互联网服务，因此，在搭建整个Online Judge时，将Web Platform服务器，Judge System 服务器以及数据库服务器搭建在同一私有局域网中，使得任何Web 请求只能在Web Platform 服务器上处理，Web Platform 服务器和Judge System服务器只通过UDP 进行问题测试数据文件的同步，并且通过数据库服务器将用户提交测试程序相关记录进行耦合。这样Judge System服务器只需要和Web Platform服务器和数据库服务器通信，因此可以通过Linux 本身提供的iptables 进行网络限制。Iptables 是通过将规则组成一个限制列表来实现绝对详细的访问控制功能，将iptables定义的规则放到内核空间当中的netfilter（网络过滤器）来读取，从而实现路由前、数据包流入口、转发管卡、数据包出口、路由后这五个关卡的控制。在本系统中，我们只需要限制Web Platform 服务器和数据库服务器的进出即可，这样可以限制网络通信，避免可能的信息泄露。

### 5.2.6记录系统资源

Linux内核提供了一个系统函数，wait4()。其主要功能是：等待子进程结束，并获取该子进程使用的系统资源（struct rusage \*rusage）及其状态（int \*status），其中struct rusage包括了使用的CPU时间，以及最后释放的内存页数；status是子进程结束的状态，通过status就可以知道子进程是如何结束的，是超出了系统资源限制，还是非法中止。

Wait4()，可以满足测评系统大部分的要求，可是却有一点无法满足。就是精确测量程序最大使用的内存。而且若程序非法中止的时候，wait4()更是完全没有办法获取内存使用数。

为了更好的解决精确测量程序最大使用内存数。只能是直接的从Linux的内核入手—查看proc文件系统。proc文件系统和sysfs文件系统类似，是虚拟文件系统，存在于内存中，用于内核和用户程序交互等以及查看内核信息，并且是动态获取有关信息。/proc/n中存储的则是系统当前正在运行的进程的相关信息，其中n为正在运行的进程，读取该目录下的status文件中的内容可以得到称为VmSize的内存信息，其大小表示的就是进程试图获取的内存，不管是否真正获取使用，因此这种方式得到的内存大小比wait4()得到的最后释放内存页数更加严格，更加准确。

## 5.3 用户空间相关程序的详细设计及实现

### 5.3.1 Judge任务监控

为了以后再有需要的时候可以很方便的添加多台后台Judge 服务器，用于扩充测评系统的负载能力。我们采取的是较为“软”的来接方式，即通过数据库，连接后台Judge服务器和前台Web服务器。

在后台Judge服务器中，有一个Judged进程，专门用于在后台服务器上创建监控数据库中用户提交测试记录。当查询到有用户提交测试记录时，都是先将其保存到共享内存中的一个等待队列上面，再利用资源信号控制量去尝试占有running的资源（根据后台服务器的实际配置，定出一个允许同时进行测试的程序的上限制）。若获得running资源则创建具体的Judge\_client 处理子进程，对该提交程序进行测试。若无法获得running资源，则自行进入阻塞状态，等待有running资源被释放的时候才进行测试该提交程序。

### 5.3.2 处理用户程序

该（Judge\_client）进程是Judged进程的子进程，其主要功能如下：

1. 当该子进程被创建后，它将参数中取到用户提交测试记录的编号，再去连接数据库，通过提交记录编号获取需要进程测试的程序的一些基本信息，包括提交测试程序的源代码；使用的是什么程序设计语言；是哪道题的源程序；是否需要采用特殊程序来对其输出进行裁决；最多允许使用多大的内存空间；多少的CPU处理时间等。在后台服务器，利用获取到的信息创建该程序所需要的文件夹，用于方便对测试程序的管理，及安全控制。
2. 根据获取到的源代码，保存到第一步所创建的目录中。
3. 对获得的源代码，根据其使用的程序设计语言，对其进行编译。若编译失败则将编译错误信息写入数据库中，并在数据库中更新该次提交的状态为编译错误。若编译成功则继续执行下一步。
4. 检查该程序对应的题目的测试数据是否已经存在后台服务器中，若不存在，则从前台服务器中同步过来。接着将测试数据文件复制到第一步所创建的目录中，并创建测试程序子进程。

### 5.3.3 运行测试用户程序

该测试进程是Judge client进程的子进程，其主要任务是：

1. 设置该进程允许使用的最大内存空间，最多CPU处理时间，最大可以输出的文件大小，最大子进程数，最大堆栈空间。
2. 利用freopen函数将该进程的标准输入，标准输出和错误输出重定向到指定路径文件。
3. 利用chroot改变该进程的根目录，利用setuid改变该进程的user id，利用setguid改变该进程的group id。
4. 利用alarm函数，设置一个定时闹钟，在时间到达后，将向该进程发送SIGALRM信号，使之被并强行中止并被父进程跟踪到。这样我们可以防止某些待测试程序，无论其CPU运行时间是否达到了限制，只要在实际时间中，已经运行超过一定的限制就会将其中止。
5. 利用execl函数，以及父进程中的参数，将该子进程替换为待测试程序进程，并在该子进程的空间中运行。

该测试进程的父进程（Judge client）的主要任务是：

1. 不断通过调用wait4()，ptrace()等系统调用函数以及查看proc文件，获取其子进程的相关信息，包括子进程目前状态（是否正常运行，是否正常中止，是否超出文件资源限制，是否被强行中止），累计使用的CPU处理时间，使用的最大内存空间等，同时也查看是否进行了非法系统调用，若是则直接中止该子进程。
2. 若子进程为非法中止，将其非法中止原因（包括Run time error， memory limit, cpu time limit 等各种非法中止），累计的使用CPU处理时间，使用的最大内存空间数更新到数据库该次提交记录对应的状态中去。然后转去最后一步。若子进程正常中止，则转入下一步。
3. 再次检查该程序运行的CPU时间，以及内存的使用数，判断是否超出了限制。若超出，则按照上一
4. 若题目是特殊判断则利用该目录下的特殊程序，否则利用程序中的自定义的比较函数对程序的输出进行裁定，将判定的结果更新到数据库该次提交记录对应的状态中去。
5. 释放running资源，释放等待队列资源。

# 6 总结

本系统在功能上主要实现了网站后台和测评系统后端服务。通过基于Django框架下的网站后台设计来保证网站的并发访问和安全服务。在测评系统后端服务中则通过设计出Sandbox机制来达到提高服务整体的安全性能的目的，并通过Linux内核proc虚拟文件系统来精确测量测评程序最大使用内存空间。但是本系统在功能上还是有很多不足的地方，比如：测试程序的正确率计算，支持多种程序设计语言等。本系统也会随着以后学校在ACM/ICPC竞赛中的发展而继续开发、更新更多功能。

**【参考文献】**

[1] Jeff Forcier, Paul Bissex, Wesley Chun. Python Web Development with Djang[M]. Addison-Wesley Professional, 2008.11

[2] W. Richard Stevens, Stephen A. Rago. Advanced Programming in the Unix Environment. Addison-Wesley Professional, 2005.06

[3] 刘楠. ACM在线评判系统设计与实现[D] 浙江：浙江工业大学，2010.2 1-5

[4] 张海涛. Online-Judge System评判子系统设计与实现[D] 哈尔滨：哈尔滨工程大学，2009 2-4

[5] 何迎生. Online Judge评判内核的设计与实现[D] 湖南：吉首大学，2010.9 3-4

[6] 鲁静轩. 程序在线评测系统的设计与实现[D]吉林：北华大学，2008 5-6

[7] Django Software Foundation. The web framework for perfectionists with deadlines.

https://www.djangoproject.com

[8] Linux Kernel Travel. Linux kernel system calls.

http://www.kerneltravel.net/journal/iv/syscall.htm

[9] Wikipedia. ACM International Collegiate Programming Contest. http://en.wikipedia.org/wiki/ACM\_International\_Collegiate\_Programming\_Contest

[10] Wikipedia. Django. http://en.wikipedia.org/wiki/Django

# 致谢

衷心地感谢张小健老师在我做的毕业设计的过程中给予的悉心指导！感谢学生程序设计创新基地实验室（ACM实验室），提供的高性能服务器作为测试环境，才能让我的毕业设计有实际用武之地。同时，非常感谢和我做同一个项目的另一位同学，在后台服务需要进行测试的时候，给予了很多支持。在此，我再次的向他们表示由衷的感谢！

# The Design，establish and development of the Shenzhe University real-time program judge system

——Design and implementation of the background web development and judge service

【Abstract】This is an article for the design and implementation of real-time online program judge system part of back-office services, including website background and judge systems background. In the Linux operating system environment, we use Django python-based framework to achieve high concurrency sites and high security access, and apply C language functions and kernel system call with providing by system to create a high-security, high stability and high accuracy evaluation backend server systems. This article describe functional design of the judge system from the requirements analysis, system design and other aspects. And then it show the part of real-time program judge system details from the website back-office functions and the judge system services functions.

**【Key words】**Web backend; Django; Online judge system; Linux kernel system calls; ACM / ICPC