OJ 大作业报告

张子颖 2021010734

一、简单的程序结构和说明

程序源码位于 src 文件夹中，主要包括服务端源代码和 src/ui 文件夹下的前端代码。

1、服务端

服务端的主体框架在 main.rs 中实现，主要是一些服务端启动前的准备工作。首先是根据命令行参数读取并解析配置文件；之后是从持久化存储中读取 Contest 列表、Job 列表和 User 列表，分别由三个全局变量（CONTEST_LIST、JOB_LIST、USER_LIST）存储，并根据 User 列表是否为空判断是否新建 root 用户；然后是新建评测线程持续获取从 channel 中传来的任务信息并运行评测过程；最后是启动服务端。

服务端的各 API 均采用同名的 async 函数实现并存储在同名 rs 文件中，比如 POST /jobs 通过 src/post_jobs.rs 文件中的 post_jobs 函数实现。

与配置文件各字段相关的结构体和枚举定义在 src/config.rs 文件中，如 Server 结构体定义了一个服务器，包括 bind_address 和 bind_port 两个字段；ProblemType 枚举定义了问题的类型，包括 Standard、Strict、Spj、DynamicRanking 四种取值。其他与请求正文和响应正文各字段相关的结构体和枚举以及他们的初始化函数 new 定义在 src/stru.rs 中，如 User 结构体定义了一个用户，包括 id 和 name 两个字段；State 枚举定义了问题的当前状态，包括 Queueing、Running、Finished、Cancled 四种取值。

评测过程由在 src/evaluate.rs 中定义的 evaluate 函数实现，evaluate 函数接收四个参数：Problem 类型的结构体的不可变引用（该结构体定义了一个问题，各字段与配置文件中的 problems 字段中的元素相同）、Submiss 类型等结构体的不可变引用（该结构体定义了一个提交，各字段与 POST /jobs API 的正文相同）、Language 类型的结构体的不可变引用（该结构体定义了一个编程语言，各字段与配置文件中的 languages 字段中的元素相同）和所要评测的任务在全局变量 JOB_LIST 中的下标 index。该函数通过前三个结构体参数提供的信息运行评测，并根据 index 实时更新 JOB_LIST 中的相关数据。该函数没

有返回值。

2、前端

前端代码位于 src/ui 的三个 html 文件中。其中 index.html 为主页面，post_jobs.html 为提交任务界面，get_ranklist.html 为获取排行榜界面。在主界面中分别点击 Submit a program 或 Get ranklist 链接可以分别跳转到上述两个界面进行相关操作。具体实现在第三部分“提高要求的实现方式”中。

二、OJ 主要功能说明和截图

实现了作业文档要求的所有基础功能，实现的提高功能有 webUI、多比赛支持、基于普通 json 文件的持久化存储、非阻塞评测、DELETE /jobs/{jobid} API、打包测试、Special Judge。

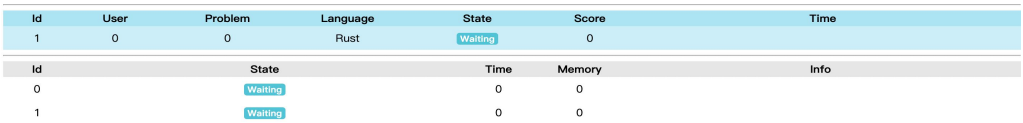
webUI 可以提交程序和获取比赛排行榜。

在主界面点击 Submit a program 或 Get ranklist 链接可以分别跳转到提交程序的页面和获取比赛排行榜的页面。如图：

在提交程序页面，通过输入有关信息可以提交程序并获得评测结果。

未提交程序时的界面：

等待运行评测时的界面：



正在运行评测时的界面：

Details

Id	User	Problem	Language	State	Score	Time
18	0	1	Rust	Running	0	

Id	State	Time	Memory	Info
0	Running	0	0	
1	Waiting	0	0	
2	Waiting	0	0	
3	Waiting	0	0	
4	Waiting	0	0	
5	Waiting	0	0	
6	Waiting	0	0	

评测运行完成后的界面：

Details

Id	User	Problem	Language	State	Score	Time
9	1	1	Rust	Accepted	100	2022-09-08T12:46:25.393Z

Id	State	Time	Memory	Info
0	Compilation Success	395230	0	
1	Accepted	287880	0	
2	Accepted	2485	0	
3	Accepted	2841	0	
4	Accepted	1825	0	
5	Accepted	1715	0	
6	Accepted	2081	0	
7	Accepted	1973	0	
8	Accepted	1810	0	

同时为了更加贴近 TUOJ 的体验，当鼠标指针悬停在某一数据点所在行时，该行的背景颜色会加深。

在获取排行榜页面，通过设置比赛 id、scoring rule 和 tie breaker 可以获取比赛排行榜：

Ranklist

Contest Id: 0
Scoring Rule: Latest
Tie Breaker: Submission Time
GET

Ranklist

#	Id	Name	Total Score	Detailed Scores
1	0	root	0	0 0
2	1	user1	0	0 0
2	2	user2	0	0 0

同时为了更加贴近 TUOJ 的体验，当鼠标指针悬停在某一用户所在行时，该行的背景颜色会加深。

其余提高功能均能实现文档要求的功能。其中非阻塞评测功能会在收到评测请求时立即返回，然后使用额外的线程运行评测并实时更新评测状态（从上面 webUI 评测结果变化的截图中可以看出）。

三、提高要求的实现方式

1、webUI

UI 功能基于 React 框架渲染排行榜和评测结果。

当用户通过 UI 提交程序时，javascript 脚本根据用户输入的信息向服务端发送 POST /jobs 请求，根据响应的 json 数据渲染评测结果。由于实现了非阻塞评测，所以所有的数据点的结果均显示为 Waiting。此时点击“refresh”按钮会触发 javascript 脚本向服务端发送 GET /jobs/{jobid} 请求，并根据获得的 json 数据重新渲染评测结果。

当用户通过 UI 获取比赛排行榜时，javascript 脚本根据用户输入向服务端发送 GET contests/{contestid}/ranklist 请求，并根据响应的 json 数据渲染评测结果。

2、持久化存储

持久化存储采用基本的 json 文件实现。`JOB_LIST`、`CONTEST_LIST` 和 `USER_LIST` 分别存放在 src/data 下的 job.json、contest.json 和 user.json 中。当程序启动时，将这三个文件中的数据分别读入到三个全局变量中（若命令行参数中包括--flush 参数则会先将文件清空）。在可能使这三个全局变量发生变化的函数（如评测函数 evaluate）中，会将可能发生变化的变量重新写入到对应的 json 文件中，以保证对应文件可以获得及时更新。

3、非阻塞评测和 DELETE /jobs/{jobid} API

非阻塞评测基于 channel 实现。将绑定在 channel 上的 producer 由服务端随配置数据传送给 API。以 POST /jobs API 为例，该 API 获取用户提交评测任务的请求后会新建并初始化一个评测任务，并将其加入到 JOB_LIST 中，随后将 `evaluate` 函数运行评测时所需的参数包装后通过复制的 producer 发送到 channel 的缓冲区，之后迅速将新建的 job 响应给用户而无需等待评测运行完成。在 main 函数中新建评测进程，通过循环使 consumer 持续监测 channel 并从中接收数据交给 `evaluate` 函数运行评测。

当用户发起 DELETE /jobs/{jobid}的请求时，API 会在验证请求合法后将 JOB_LIST 中对应的 job 的 `result` 更改为 `Canceled`，`state` 更改为 `Finished`。当 producer 发现接收到的 `index` 所对应的 job 的 `state` 是 `Queueing` 时才会将参数传给 `evaluate` 函数运行评测，否则将直接跳过该 job。因此被发送 DELETE 请求的 job 会跳过评测过程。

4、打包测试

为了统一打包测试和普通的测试，在对各测试点运行之前会先对测试点进行分组。若为打包测试则按 problem 中指定的方式分组，否则每一题分为一组。在评测过程中遍历每一组，在对一组测试点进行评测时，定义 bool 类型的变量 is_group_ok 并初始化为 true 来记录当前组是否已有错误的测试点。然后遍历该组中的所有测试点，若 is_group_ok 为 false 则直接将结果置为 Skipped 并转向下一个测试点；否则评测该测试点，若出现错误则将 is_group_ok 置为 false。当一组评测完成时，若 is_group_ok 仍为真则在整题得分中加上该题的得分，否则不加分。

5、Special Judge

运行可执行文件后将获得的输出文件和答案文件传给题目指定的评测脚本进行评测，之后获取评测脚本的输出文件。若输出文件的格式不符合预期则将该点置为 Spj Error，否则将该数据点的 result 字段置为输出文件的第一行，info 字段置为输出文件的第二行。

四、完成此作业的感想

1、对 web 的知识有了更加系统的认知。最早是在学习爬虫的过程中对浏览器如何通根据网络响应生成我们看到的页面获得了对 web 的基本了解；这次大作业自己搭建一个服务端和简易的前端，对 web 有关知识的认知更加系统化、深入化了。

2、代码风格得到了改善。与 wordle 大作业相比，此次大作业的代码复用率得到了显著提升；但模块划分仍然不足，比如所有实现 API 的函数均位于 src 文件夹下而没有更加细化。

3、阅读英文官方文档和示例源代码的能力仍然有待加强。在完成非阻塞评测后，我本想实现独立评测进程。但限于阅读英文文档和示例源代码但能力，没能成功使用 amiquip 库实现独立评测进程。