Computer Science and Application 计算机科学与应用, 2026, 16(9), 37-46
Published Online September 2026 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2026.169287

Hans汉斯

面向Linux系统实验课程的Online Judge平台

蔡华帅*#，孙　皓

江西理工大学信息工程学院，江西　赣州

收稿日期：2026年8月1日；录用日期：2026年8月31日；发布日期：2026年9月8日

摘　要

Linux系统实验课程具有环境配置复杂、操作路径多样、阶段状态难以人工核验等特点。传统实验批阅主要依赖报告、截图和最终文件，难以还原学生真实操作过程；传统Online Judge又更适合程序输入输出比对，难以直接评价文件权限、进程状态、服务运行和脚本执行等Linux实验任务。为解决上述问题，本文设计并实现了一套面向Linux系统实验课程的Online Judge平台。平台基于Docker提供统一隔离的在线实验环境，采用WebSocket与伪终端技术实现浏览器端命令行交互，使用检查点式评测脚本对阶段任务进行自动判分，并通过终端回放、文件变化记录和AI辅助批阅形成可复核的过程证据链协助教师完成实验效果评估。

关键词

Linux系统实验，Online Judge，Docker容器，终端回放，AI辅助批阅

Online Judge Platform for Linux System Experiment Courses

Huashuai Cai*#, Hao Sun

School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou Jiangxi

Received: August 1, 2026; accepted: August 31, 2026; published: September 8, 2026

Abstract

Linux system laboratory courses are characterized by complicated environment configuration, diverse operation workflows, and difficulties in manual verification of intermediate execution states. Traditional experimental grading mainly relies on laboratory reports, screenshots and final files, which makes it impossible to restore students' actual operation processes. Conventional Online Judges are more applicable to input-output comparison of programs, and cannot directly evaluate

*第一作者。
#通讯作者。

文章引用: 蔡华帅, 孙皓. 面向 Linux 系统实验课程的 Online Judge 平台[J]. 计算机科学与应用, 2026, 16(9): 37-46.
DOI: 10.12677/csa.2026.169287

Linux experimental tasks involving file permissions, process statuses, service operation, script execution and other metrics. To address the above challenges, this paper designs and implements an Online Judge platform tailored for Linux system laboratory courses. Built upon Docker, the platform provides unified and isolated online experimental environments. It adopts WebSocket and pseudo-terminal technologies to realize command-line interaction on web browsers, leverages checkpoint-based evaluation scripts for automatic scoring of phased tasks. Furthermore, it generates a verifiable procedural evidence chain through terminal playback, file change logging and AI-assisted grading, assisting instructors in assessing students' experimental performance.

Keywords

Linux System Experiments, Online Judge, Docker Containers, Terminal Playback, AI-Assisted Grading

1. 引言

Linux 系统实验是计算机类专业实践教学的重要内容。学生需要在命令行环境中完成文件管理、权限设置、软件包安装、服务启动、脚本编写等连续操作，任务结果常常不是一个固定答案，而是一组系统状态、运行过程和阶段结果的综合表现。教师如果只依据实验报告、截图或最终文件给分，往往难以判断学生是否真正理解了关键命令，也难以发现学生在中间步骤中的错误和反复试探行为。

Online Judge 系统在程序设计课程中已经得到较多应用，能够通过自动运行测试用例给出快速反馈[1]。但 Linux 实验与一般编程题不同，其评价对象不仅包括程序输出，还包括文件权限、进程状态、服务端口、目录结构和脚本执行副作用。Docker 容器技术能够为在线 Linux 实验提供轻量、可复制、易重置的环境[2]。相关课程实验平台研究也表明，基于 Docker 的实验环境具有较好的可实施性[3]；在线实验平台还能降低学生本地配置虚拟机的负担[4]。基于 Web 的在线判题系统研究进一步说明，自动判题平台能够辅助程序设计类课程教学[5]；自动形成性反馈研究也表明，反馈信息设计会影响学生修改和理解问题的效果[6]。因此，将 Online Judge 的阶段反馈思想与容器化 Linux 环境结合，是改进 Linux 实验教学评价方式的可行方向。

Table 1. Platform feature comparison
表 1. 各平台功能对比

平台	实验环境	实验设计	实验执行	测评机制	AI 应用
传统方式	虚拟机，学生自行安装镜像	文本形式	交互式操作	根据实验报告评分	无
LabEx	Web 页面交互，统一的虚拟机镜像	平台预设计	交互式操作	根据配置自动执行判题脚本测评	AI 根据错误操作提供操作指导
WebVM	Web 页面交互，不可控的虚拟机镜像，浏览器存储	文本形式	交互式操作	根据实验报告评分	无
本平台	Web 页面交互，教师可配置统一的 docker 镜像，服务器存储	可分值配置，检查点配置，判题脚本	交互式操作，脚本方式，文件上传	根据配置自动执行判题脚本测评，全实验过程记录可回放	结合证据包对实验得分给出评价与建议

现有基于 WebAssembly 的 Linux 交互式平台(如 WebVM)虽能提供终端访问能力，但普遍缺乏面向实验教学的功能设计，少数具备自动测评功能的平台(如 LabEx)，则仅侧重于学习者端的操作体验，在教师端的实验管理功能上仍存在明显缺失。表 1 从 Linux 系统实验的全流程环节(实验环境、实验设计、实验执行、测评机制)及 AI 应用程度两个维度，对上述平台与本文项目进行了横向对比。

本文围绕 Linux 系统实验的过程评价需求，设计并实现面向 Linux 系统实验课程的 Online Judge 平台。平台并不把课程管理作为唯一目标，而是围绕学生在线实验、系统自动评测、过程证据采集和教师复核批阅展开。与传统批阅模式相比，本文平台将实验环境、检查点评测、终端回放、文件变化和 AI 建议纳入同一条链路，使教师能够在较低批阅负担下获得更完整的评价依据。

2. 平台设计

2.1. 设计目标

平台设计目标包括四个方面。第一，提供统一的 Linux 在线实验环境，使学生无需在本地反复配置虚拟机，教师也能围绕一致的系统版本和软件环境组织实验。第二，提供检查点式自动评测机制，将较长的实验拆分为若干阶段任务，并通过教师配置的判题脚本检查文件、权限、进程、服务或脚本运行结果。第三，保存可复核的实验过程证据，包括终端操作回放、检查点评测结果和关键文件变化记录。第四，引入 AI 辅助批阅，在自动评测结果和过程证据基础上生成建议分、评语和待关注项，但最终成绩仍由教师确认。

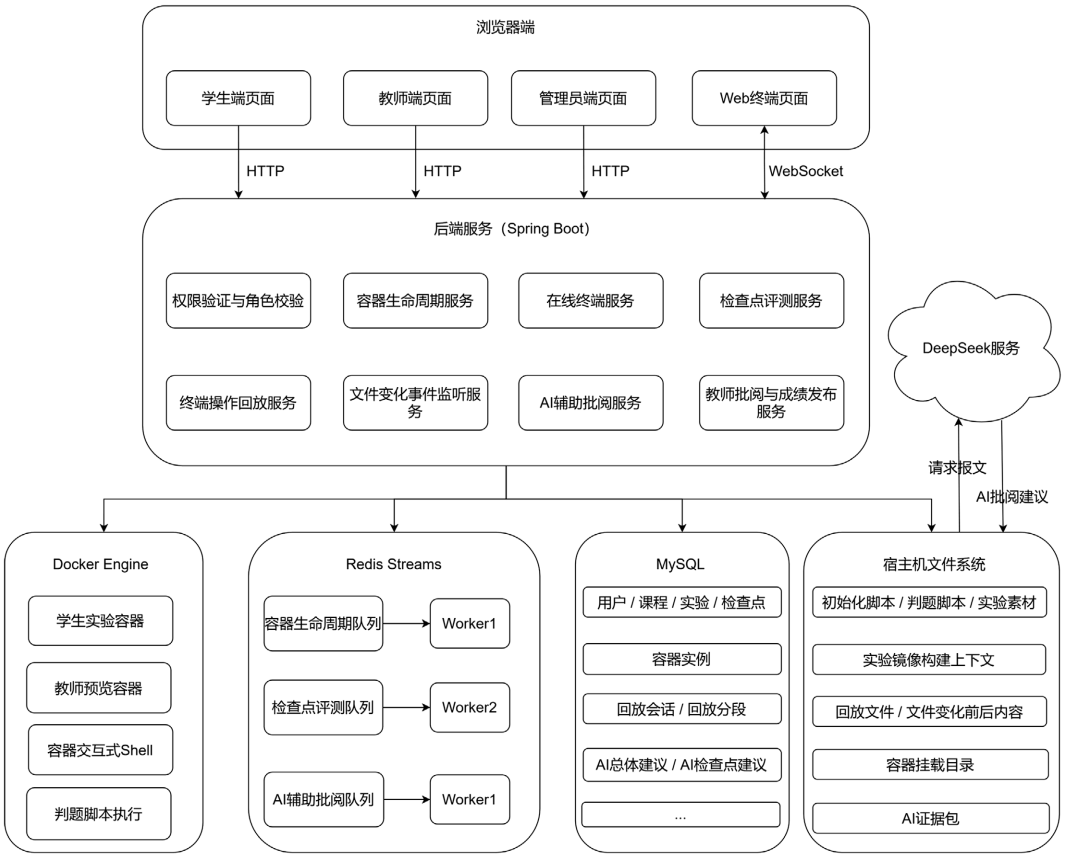

Figure 1. System overall architecture diagram
图 1. 系统总体架构图

2.2. 总体架构

系统采用前后端分离的 B/S 架构，主要由浏览器端、后端服务、Docker Engine、Redis Streams、MySQL 数据库、宿主机文件系统和外部 AI 服务组成。浏览器端基于 Vue3 实现，普通业务通过 HTTP 接口访问，在线终端通过 WebSocket 建立双向连接。后端基于 Spring Boot 实现身份认证、权限校验、容器调度、终端会话管理、评测任务处理和 AI 建议生成等业务。

Docker Engine 负责创建和管理学生实验容器。系统根据实验镜像和资源策略为学生准备隔离环境，并通过 CPU 和内存限制控制资源占用。MySQL 保存用户、课程、实验、检查点、容器实例、评测结果、回放会话、文件变化事件和 AI 建议等结构化数据。Redis Streams 用于承载容器生命周期、检查点评测和 AI 辅助批阅三类异步任务，避免耗时操作阻塞前台请求。宿主机文件系统保存实验素材、判题脚本、终端回放文件、容器挂载目录和 AI 证据包。系统总体架构如图 1 所示。

2.3. 功能结构

平台面向管理员、教师和学生三类角色。管理员负责用户与资源管理；教师负责课程创建、实验发布、题目与检查点配置、判题脚本上传、学生容器管理和成绩批阅；学生通过邀请码加入课程，在已开放实验中使用 Web 终端完成 Linux 操作，提交检查点并查看反馈。一次完整实验教学的业务流程见图 2。

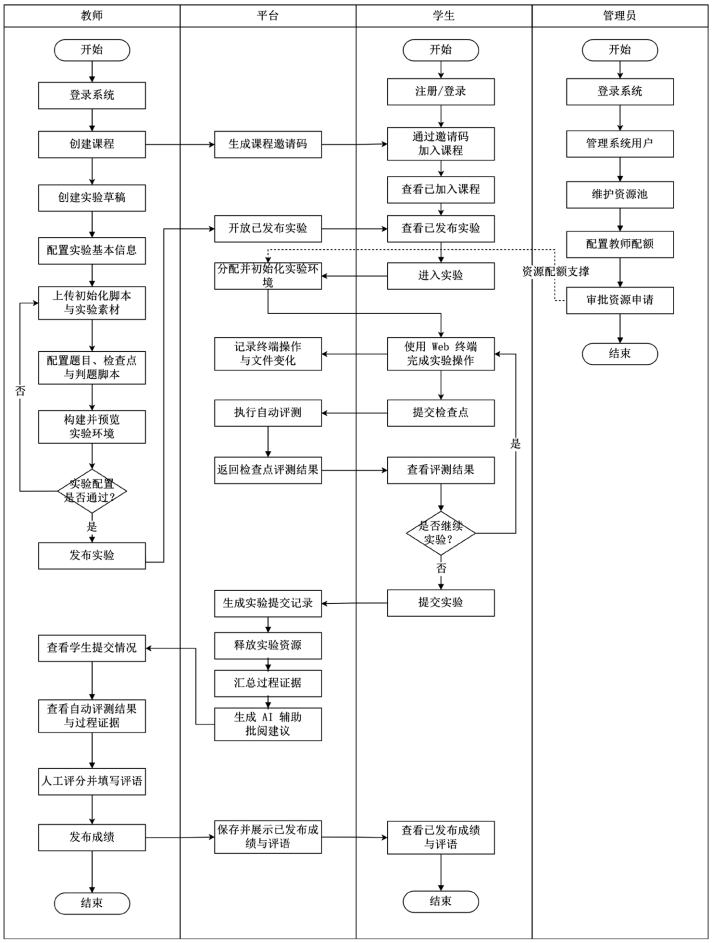

Figure 2. Schematic diagram of system business processes
图 2. 系统业务流程示意图

3. 系统实现

3.1. 容器化实验环境调度

容器化实验环境是平台运行的基础，主要使用 docker-java SDK 实现。系统为学生维护容器实例状态，主要包括 STOPPED、RUNNING 和 REBUILDING。学生进入实验时，如果不存在可用容器，系统会创建容器；如果已有容器与当前实验环境不匹配，系统提示学生切换或重置；当学生主动重置、教师干预或容器长期空闲时，系统执行停止、销毁或重建操作。学生容器生命周期如图 3 所示。

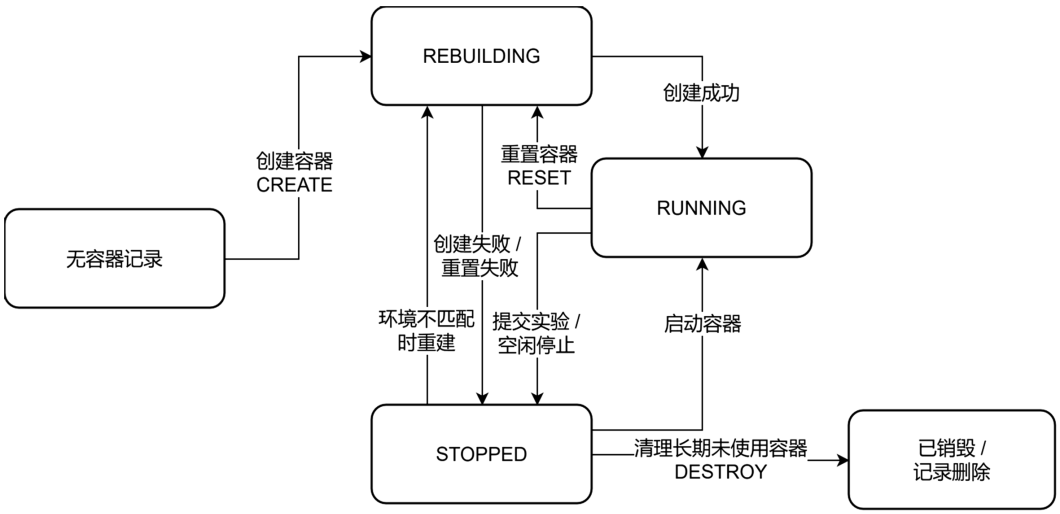

Figure 3. Student-container lifecycle state machine diagram
图 3. 学生容器生命周期状态机图

容器创建、重置、停止和销毁都可能涉及 Docker 调用、素材投放和旧连接清理，执行时间不稳定。为避免这些操作阻塞页面请求，系统将容器生命周期任务写入 Redis Streams 队列，由后台 Worker 异步消费。前端只根据数据库中的容器状态展示等待、可连接或异常提示。该设计保证了前台接口快速返回，也使容器任务能够按顺序处理。

3.2. Web 终端交互与过程记录

学生端在线终端页面如图 4 所示。页面左侧展示实验说明、题目、检查点状态和提交入口，右侧为 XTerm.js 终端区域。后端在 WebSocket 握手阶段校验 Token、实验编号和用户角色，校验通过后通过 script 命令包装 docker exec-it 启动容器内交互式 Shell。script 命令利用伪终端机制连接 Shell，使浏览器端输入能够写入容器终端，容器输出再经 WebSocket 返回前端显示。

终端交互同时承担过程证据采集任务。后端在转发容器输出时，将输出内容写入 io 文件，并按输出间隔和字节数写入 timing 文件。教师批阅时，前端播放器读取这两个文件并按时间轴还原终端显示过程。相比只保存命令文本，终端回放可以保留全屏程序、报错输出、编辑器界面等更丰富的信息，为教师判断学生是否真实完成操作提供依据。已有浏览器端 Linux 环境研究表明，Web 化实验环境能够降低学习者部署 Linux 环境的门槛[7]。将实验终端的完整输入输出过程予以全量记录，不仅是实现 Web 化操作的前提，更使得底层系统配置管理得以与上层实验业务流程彻底解耦，从而为后续扩展(如适配更多 Linux 发行版)提供了充分的灵活性。

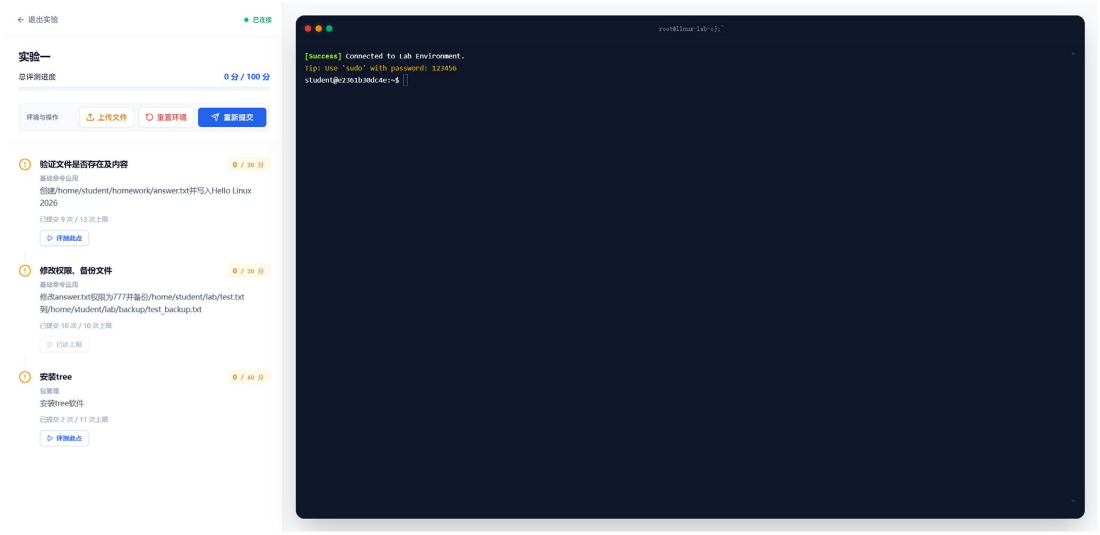

Figure 4. Student-side online terminal
图 4. 学生端在线终端页面效果图

3.3. 检查点自动评测

Linux 实验通常由多个阶段任务组成，例如创建文件、修改权限、启动服务、编写脚本和验证输出。平台将实验内容组织为题目和检查点，每个检查点可以配置多个判题脚本及其分值、超时时间和提交次数上限。学生完成阶段任务后提交对应检查点，系统先记录评测任务并入队，后台 Worker 再进入学生容器执行脚本，最后将脚本通过情况、得分、失败原因和评测状态写回数据库，时序图如图 5 所示。检查点式评测使学生能够在实验过程中获得阶段反馈，也使教师能够定位学生在哪一步出现问题。

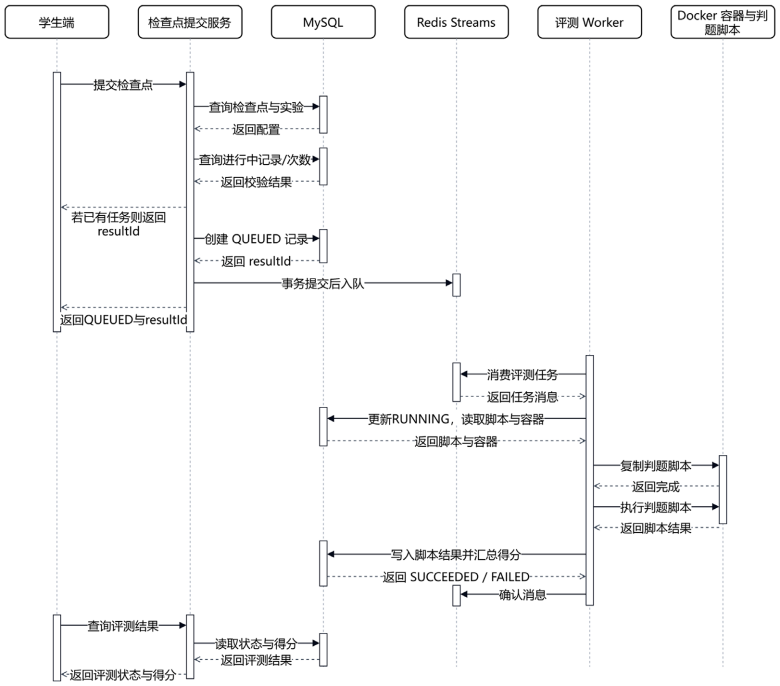

Figure 5. Checkpoint assessment timing diagram
图 5. 检查点评测时序图

自动评测流程不可避免地引入如拒绝服务攻击、提权入侵宿主机等安全风险。考虑到平台需要广泛支持各类 Linux 系统实验，若对容器内的系统调用施以严格限制，可能影响部分实验的正常执行，因此未采用常见的系统级调用控制方案。针对上述风险，平台针对可能的攻击行为在应用层设计了一定的防御机制：平台为检查点提供提交次数限制(如图 6 所示)，并在教师批阅时展示提交次数和历史结果，便于识别异常行为；在判题执行层面，判题脚本由后端服务临时注入 Docker 容器的特定目录运行，执行结束后即自动销毁，而返回码获取、文件拷贝、结果判定与数据落库等关键操作均在容器外部的后端服务中完成，确保评测逻辑与执行环境的隔离。此外，平台还结合文件监听机制，可有效识别学生在评测过程中是否尝试窃取判题脚本逻辑以硬编码答案，进一步保障了评测的公平性。

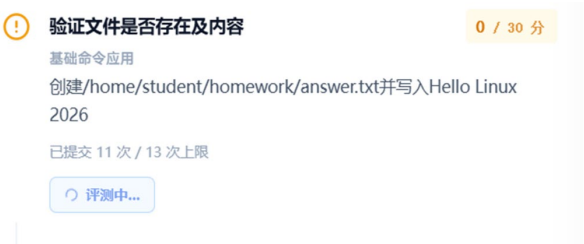

Figure 6. Checkpoint online judging status transition
图 6. 检查点评测状态变化效果图

3.4. 实验过程证据采集

仅依赖自动评测结果仍然不足以评价 Linux 实验过程。学生可能通过偶然操作得到正确状态，也可能在最终状态正确的情况下经历大量无效尝试。为此，平台将自动评测结果、终端回放和文件变化记录共同组织为过程证据链，如图 7 所示。

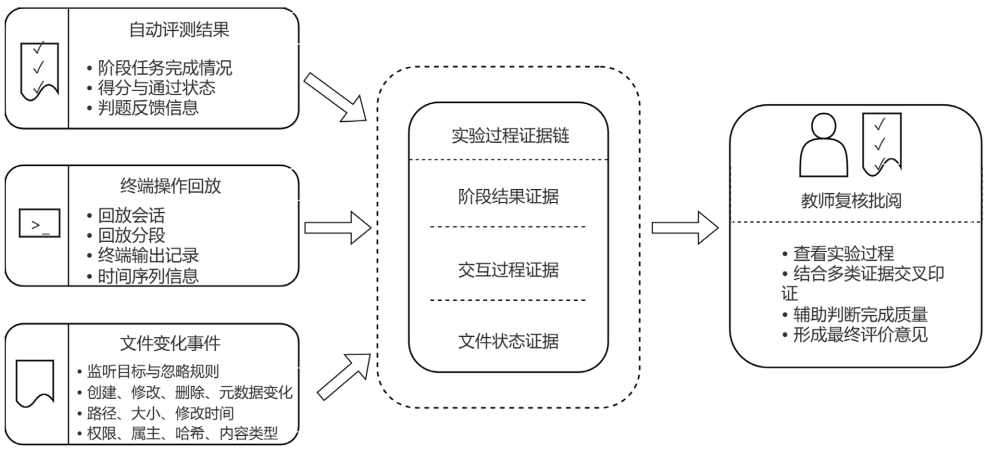

Figure 7. Experimental process evidence chain composition diagram
图 7. 实验过程证据链组成图

文件变化记录基于容器目录挂载和目录监听，使用 maven 仓库中的第三方库 io.methvin.directory-watcher 实现。教师在实验配置中指定需要监听的容器内路径，系统将对应宿主机挂载目录作为监听对象。学生终端连接建立后，后端启动监听服务并建立基线快照；后续文件创建、修改、删除以及权限、属主、时间等元数据变化会被记录为事件，具体流程如表 2 所示。对于短时间内大量产生的文件事件，系统进行聚合展示，减少教师批阅时的信息噪声。

Table 2. File change monitoring and batch file event aggregation process
表 2. 文件变化监听和批量文件事件聚合流程

文件变化监听流程		批量文件事件聚合流程	
输入：文件系统事件 event，监听目标 target，快照缓存 snapshotCache 输出：文件变化事件 changeEvent 或空结果		输入：文件变化事件 event，聚合缓冲区 buffer，时间窗口 window，批量阈值 threshold 输出：普通事件或批量事件记录	
1	Path ← 将 event 路径映射为容器内路径	1	if event 命中忽略规则 then
2	if path 不属于 target 监听范围 or path 命中忽略规则或临时文件规则 then	2	return null
3	return null	3	end if
4	end if	4	key ← 取 event.path 所属目录作为聚合键
5	before ← snapshotCache[path]	5	将 event 加入 buffer[key]
6	after ← 读取 path 当前快照	6	删除 buffer[key]中超过 window 的旧事件
7	if before = null and after ≠ null then	7	if buffer[key]中事件数量 < threshold then
8	type ← CREATE	8	return event
9	else if before ≠ null and after = null then	9	end if
10	type ← DELETE	10	mainType ← 统计 buffer[key]中数量最多的事件类型
11	else if 内容哈希或文件大小发生变化 then	11	count ← 统计 buffer[key]中事件数量
12	type ← MODIFY	12	batchEvent ← 创建批量事件
13	else if 权限、属主、属组或修改时间发生变化 then	13	batchEvent.path ← key
14	type ← METADATA	14	batchEvent.eventType ← mainType
15	else	15	batchEvent.isBatch ← true
16	return null	16	batchEvent.batchEventCount ← count
17	end if	17	batchEvent.contentKind ← SKIPPED
18	根据 after 更新或删除 snapshotCache[path]	18	清空 buffer[key]
19	changeEvent ← 构造包含 type、path、before、after 的事件	19	return batchEvent
20	return changeEvent		

3.5. AI 辅助批阅

AI 辅助批阅不是替代教师给分，而是基于过程证据为教师提供提示。学生提交实验后，系统根据实验说明、检查点配置、自动评测结果、终端摘要和文件变化记录构造证据包，调用大语言模型生成建议总分、检查点建议、总体评语、置信度和待关注项。为降低模型输出不稳定带来的风险，后端对返回结果进行格式校验、分数范围校验和检查点编号校验，异常结果不会直接进入可参考状态。

教师批阅页面将 AI 建议与自动评测、文件变化和终端回放放在同一页面中展示，如图 8 所示。教师可以据此快速发现两类情况：一类是学生虽然得分较高，但存在大量无效命令、反复试探或疑似针对判题脚本构造结果；另一类是学生操作过程合理，但自动评测因环境或脚本问题给出偏低分数。最终成绩仍由教师结合证据确认。

Figure 8. Preview of teacher grading page
图 8. 教师批阅页面效果图

4. 系统测试

4.1. 测试环境与功能测试

为验证系统功能和性能，本文在统一软硬件环境下开展测试。测试设备为联想 82UU 笔记本，处理器为 AMD Ryzen 7 6800HS@3.2GHz，内存为 16GB LPDDR5 6400MHz，宿主操作系统为 Windows 10 Pro 22H2，项目运行环境为 WSL Ubuntu 22.04，浏览器为 Google Chrome。后端运行在 Java 11 和 Spring Boot 2.7 环境中，前端采用 Vue3，数据库为 MySQL 8.0，Redis 版本为 7.2，容器环境为 Docker Desktop。

功能测试分别使用管理员、教师和学生账号完成，覆盖用户认证与角色权限、课程组织、实验配置、题目与检查点配置、容器环境管理、Web 终端交互、检查点自动评测、过程证据采集、实验提交与教师批阅、AI 辅助批阅，以及未登录访问、越权访问、容器异常、终端断连和 AI 返回格式错误等异常场景。测试结果显示，各项功能均能按预期完成。

4.2. 性能测试

性能测试采用 Node.js 脚本模拟学生使用流程，设置 10、20、30 名学生三种并发规模，学生容器规格为 0.5 CPU 核和 128MB 内存。测试分为模拟真实使用场景和极限压力场景。前者模拟学生分散进入实验、读题、操作终端和提交检查点的节奏；后者去除等待间隔，使多个学生连续触发容器创建、重置、终端连接和检查点评测任务。主要指标包括普通接口 P95、容器就绪等待 P95、容器重置完成 P95、评测完

成 P95 和终端连接 P95。

在模拟真实使用场景下，系统普通接口响应稳定，30 并发时普通接口 P95 最大值为 91 ms，提交评测 P95 为 39 ms，最终提交实验 P95 为 60 ms。长耗时操作也保持在秒级范围，30 并发时终端连接 P95 为 3992 ms，评测完成 P95 为 1271 ms，容器就绪等待和容器重置完成 P95 分别为 2933 ms 和 3135 ms。测试结果如图 9 所示。

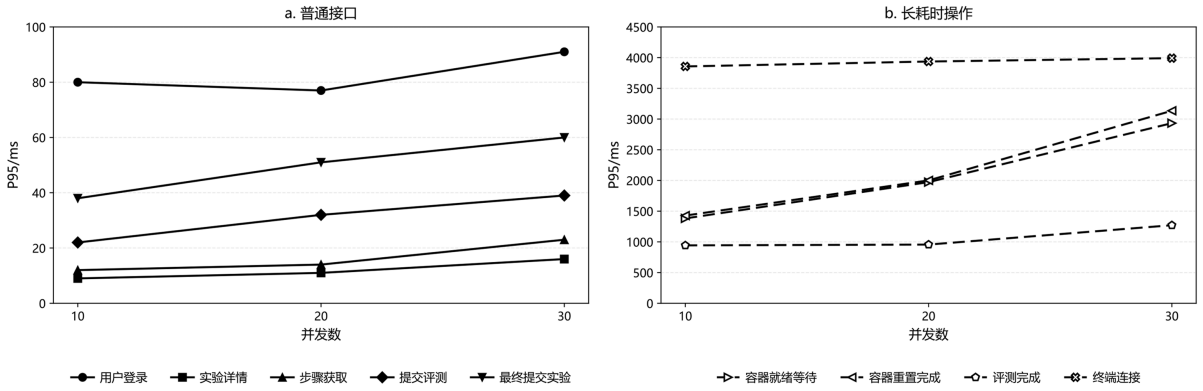

Figure 9. P95 latency line chart for critical operations
图 9. 关键操作 P95 耗时折线图

5. 小结

本文面向 Linux 系统实验课程，设计并实现了一套支持在线实验、检查点评测、过程记录和教师复核的 Online Judge 平台。平台使用 Docker 提供统一隔离的实验环境，通过 WebSocket 和伪终端实现浏览器端 Linux 操作，使用检查点脚本对阶段任务进行自动评测，并结合终端回放、文件变化记录和 AI 辅助建议形成过程评价依据。测试结果表明，系统主要功能运行正常，普通接口响应迅速，容器、终端和评测任务能够稳定完成，具备面向实际课程使用的应用价值。

后续工作可从四个方面继续完善：一是提供检查点模板化配置能力，降低教师编写判题脚本的成本；二是增强过程证据检索能力，支持按命令、错误信息、文件路径和提交时间快速定位关键操作；三是探索 AI 在实验设计阶段的辅助作用，帮助教师生成检查点和易错提示；四是细化容器资源调度策略，使平台在多课程、多实验同时运行时保持更好的稳定性。

参考文献

[1] 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): 170-177.

[2] 邱建新. 基于 Docker 容器技术的 Linux 在线实验环境设计[J]. 信息技术, 2022(2): 48-52+58.

[3] 崔艳敏. 基于 Docker 的课程实验平台设计以及实现探讨[J]. 中阿科技论坛(中英阿文), 2020(3): 144-145.

[4] 安海兵, 祁雷. 软件工程中的 Linux 操作系统实验课程教学实践[J]. 集成电路应用, 2021, 38(6): 192-194.

[5] 蔡崇超. 基于 Web 的在线判题系统设计与实现[J]. 软件导刊, 2016, 15(3): 107-109.

[6] Hao, Q., Smith IV, D.H., Ding, L., Ko, A., Ottaway, C., Wilson, J., *et al*. (2021) Towards Understanding the Effective Design of Automated Formative Feedback for Programming Assignments. *Computer Science Education*, **32**, 105-127. https://doi.org/10.1080/08993408.2020.1860408

[7] Sharrock, R., Angrave, L. and Hamonic, E. (2018) WebLinux: A Scalable In-Browser and Client-Side Linux and IDE. *Proceedings of the 5th Annual ACM Conference on Learning at Scale*, London, 26-28 June 2018, 1-2. https://doi.org/10.1145/3231644.3231703