基于分布式评测技术的操作系统实验系统

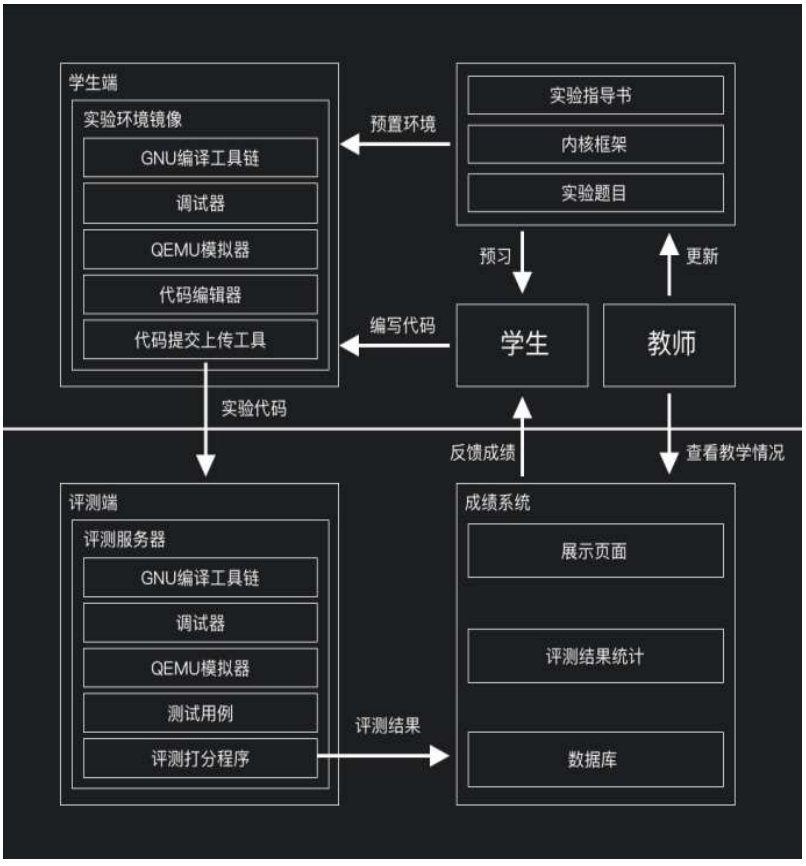
A.作品综述

作品设计、发明的目的和基本思路、创新点、技术关键和主要技术指标	**作品设计目的：**为普通高校的操作系统实验服务，为了让学生在真正的操作系统内核代码中实验，从设计角度深入理解操作系统原理，此外为了方便学生提交作业并及时查看实验结果，让老师更好地了解学生学习情况，并减轻老师实验验机的负担，我们设计了一套基于分布式自动评测技术的操作系统实验教学方案。 **基本思想：**以 xv6 内核作为骨架，参考国内外各个大学的操作系统教学方案，结合普通高校学生特点，设计不同难度实验题目让不同层次学生能力得到体现，并在操作系统实验设计评测程序，方便教师验机和学生提交作业。 **创新点：**1)采用分布式评测技术，评测端工作机可以被灵活地部署在多台计算机上，从而达到在服务器设备资源有限的情况下，灵活而又有效地满足高并发场景下评测需求。2)让学生基于真正的操作系统内核代码进行实验，而不是传统的模拟实验。

技术关键和主要技术指标：

实验平台分学生端和评测端两部分。对于评测端，在学校架设 Linux 服务器，上面搭建实验所需的编译环境（GNU 工具链和 QEMU 模拟器）。学生可以通过上传代码的方式，将代码提交至服务器。 评测程序在服务器中的编译环境下，编译学生提交的代码，将编译生成的内核加载至 QEMU 模拟器。评测程序通过向 QEMU 的标准输入流写入测试命令来测试学生所实现的功能的 完成情况（可以分为多个评测点）

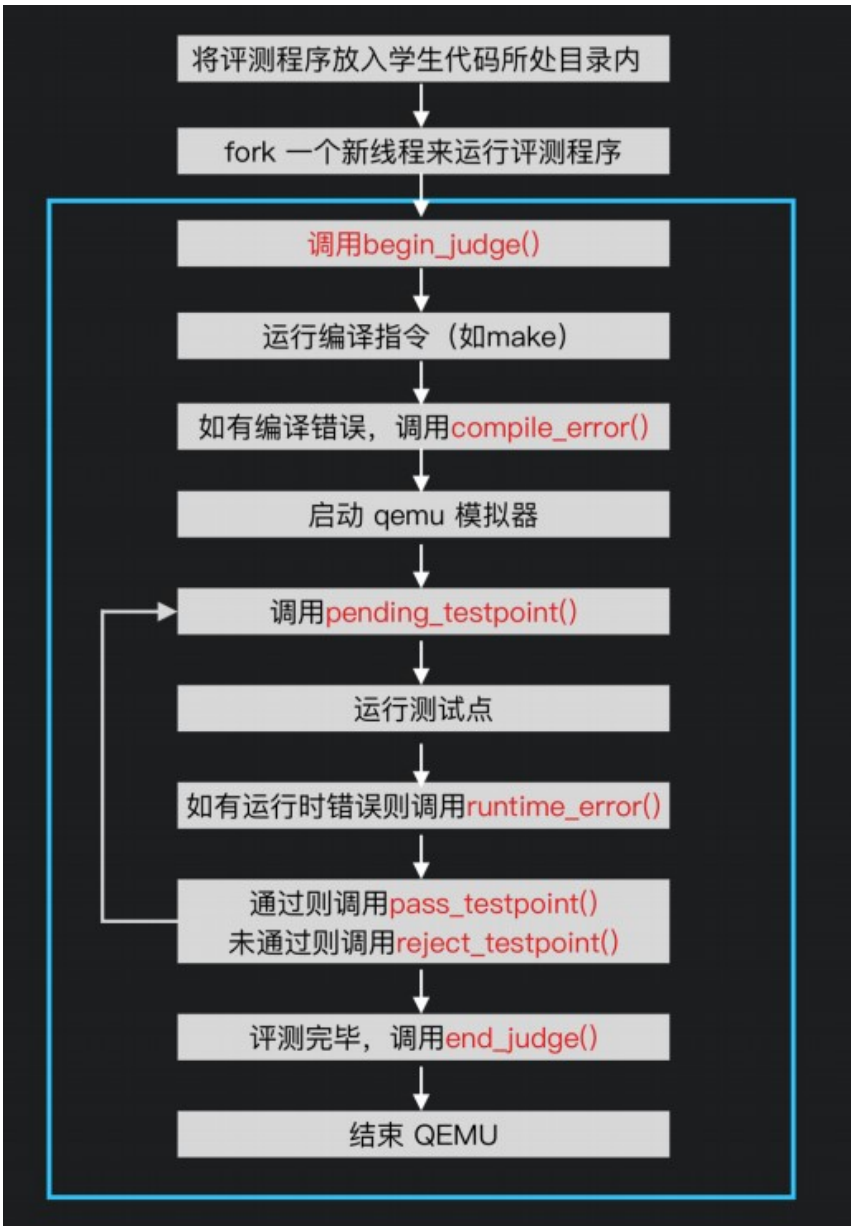
整体架构与流程图如下：



	评测系统设计：最终评测程序将输出评测结果（每个测试点的完成情况及其对应的分数、总成绩）给成绩 管理系统。 评测规则可以参考 OI 赛制，分成多个评测点，每通过一个评测点即可获得一定的分数，最 终将所有得分加在一起作为本次实验的总成绩。学生可以多次提交，通过查看评测点的完成情况，来获得及时的反馈，从而修改自己的程序以适配各种 edge case。评测端采用 QEMU 作为模拟器，GNU 工具链作为编译环境，Linux 系统（CentOS、 Ubuntu 等）作为宿主机来运行评测环境，Python 或 Bash 等脚本语言来编写评测程序。 评测端编译环境应与学生端内的编译环境完全一致，系统、编译器和模拟器的版本应严格 对应，以期达到评测端和学生端内运行同一段代码时，能获得同样的运行结果，从而达到公平公正、反馈明确的评测效果。评测机和成绩系统之间通过 kafka 消息队列进行通讯。每个用户每次提交，将会生成一个评测过程。各个模块分别负责评测过程中的各个部分。成绩系统用来处理用户提交的代码并分发评测任务，评测任务将被成绩系统提交至 kafka 消息队列，评测

	机订阅评测任务有关的主题，从而消费评测任务。评测机在消费评测任务后，同步代码文件和相关依赖至本机，并运行评测程序来对学生提交的代码进行评测。评测程序负责对评测点进行评测打分并返回评测结果给评测机。最终，评测机将评测结果通过 kafka 消息队列送回至成绩系统，成绩系统对打分结果进行汇总，并持久化存储至数据库中。而评测机程序以微服务的形式存在，可以被灵活地部署在任何装有 JDK 的宿主机或者容器中。评测机的数量可以按需调整，弹性扩容，当评测服务的并发需求较大时，只需在一台新的设备上启动预先写好的评测程序，输入评测系统主服务器的 ip 地址，评测机程序将能自动完成服务注册的功能，向主机登记当前机器的状态与承载能力。而主机在分配评测任务时，也能依据评测机的当前状况来灵活分配评测任务，将任务平均分配至各个评测机，以实现均衡负载的需求。当所有评测机执行的评测任务数量到达其任务数上限时，新的评测任务将会被送至队列等待后续消费，等待队列中的评测任务依照先来先服务的原则，在评测机有空闲任务槽位时，主机还会自动将最新来的评测任务分发至评测机。

评测程序流程图如下：



学生端设计：学生端用来为学生提供编写和调试代码的环境，以期节省学生搭建环境的时间成本，降低学生实验初期准备工作的难度，降低学生在初期时出现畏难情绪的概率。 实验环境中应有与实验题目对应的交叉编译环境（如

	GNU 编译工具链）、用于测试的模拟器（如 QEMU）和调试器（如 GDB），同时也要预置有上传代码的程序，用于学生提交代码至评测端，以通过评测结果来获得及时反馈，从而修正代码中潜在的问题。在学校的服务器上安装 Linux 操作系统，并在其中安装实验环境作为共享文件，然后为每个学生创建一个用户，学生可以通过 SSH 或 Telnet 来远程连接服务器，通过 SFTP 上传自己编写的代码，然后在服务器上编译并查看运行效果，出现问题了也可以通过 GDB 来进行调试。学生在自己的电脑上可以通过 VSCode 等代码编辑器来编写代码，然后利用 VSCode Remote 功能 来连接服务器进行远程代码编译和调试，这样可以省去学生学习使用 SSH 和 SFTP 的学习成本 （VSCode Remote 可以自动实现文件同步功能）。MIT 的实验也采用类似的方案，在服务器上搭建编译环境供学生远程连接使用。这种方案有一定的条件限制，由于实验课时有近 100 个学生同时远程连接服务器，并进行代码的编译和调试，计算量和 I/O 占用都极大，需要服务器有较高的性能以负载性能消耗，或是采用更多的服务器来分担不同批次的学生。这就意味着这个方案所需的金钱成本比较大，所以

	我们需要有相应条件。所需技术如下， Linux 系统，作为环境的基础，采用 CentOS。GNU 工具链，根据内核代码的体系结构（riscv），选择适合的 GNU 工具链，用以搭建交叉编译环境，用来编译学生编写的内核代码。QEMU，开源模拟器，用来在当前的系统中提供一个虚拟的环境，来运行学生编译生成的操作系统内核。 VSCode，开源代码编辑器，有自动补全和参数提示功能，用来方便学生进行内核代码编写。 **成绩管理与展示系统功能：**学生可以通过本系统来查看自己在每个实验中，所提交的程序代码的评测结果，以及获得分数的情况。学生可以通过本系统获得反馈，发现自己程序是否存在一些问题，以此来优化自己的程序实现。在学期末，系统可以自动导出每个学生的成绩（例如输出一个 Excel 表格），教师可以直接根据这个表格来为学生登记实验成绩。实验平台系统整体采用前后端分离架构。Web 前端负责界面渲染，将后端传来的成绩信息和实验信息等渲染成 Web 界面，以向学生展示相关的信息。 前端采用 Vue.js 框架进行开发，采用 JavaScript、HTML、CSS 作为开发语言。后端

	Web 后端负责从评测程序中获取评测结果，然后将评测结果汇总并写入数据库。同学和教师在需要查看自己的评测结果和成绩时，后端向前端传去相关信息。后端采用 SpringBoot 框架进行开发，Java 作为开发语言，MySQL 作为数据库。前后端通过 Restful HTTP API 进行通信，前后端分离开发，并共同部署在服务器中。 **实验题目设计：**阅读 xv6、ucore 等内核代码，根据教学大纲，设计实验题目，同时也包含传统的操作系统功能模拟题目，以便于不能层次水平的学生使用。例如让学生实现进程调度、存储管理、文件系统等具有代表性的模块。题目设计遵照循序渐进的原则，由最开始的搭建实验环境，再到一些简单的环境，再到具体实验模块的实现。在实验题目设计后，针对题目进行测试用例的编写，以供评测程序对学生代码进行功能实现验证和打分。 本操作系统实验平台的评测端可以在 300 名学生同时提交评测任务时稳定运行。

作品的科学性、先进性（必须说明与现有技术相比该作品是否具有突出的实质性技术特点和显著进步。请提供技术性分析说明和参考文献资料）	**作品的科学性、先进性：** 本操作系统实验解决方案为本校自主研发，贴合普通高校的课程大纲教学情况，难度适中，易于学生掌握。学生在操作系统实验的学习过程中，可以将所学的课本上的理论知识，运用到实际的工程实践中去，从零起步，循序渐进，以系统能力培养目标为导向，实验题目为主线，逐步实现一个简单而又功能完整的操作系统内核。 相较于国内外其他大学的操作系统教学方案，我们的方案更贴合我校的实际教学情况，门槛低，上限高，满足针对各个层次学生的教学需求，不同层次的学生通过我们的实验教学，都可以学到有用的知识，有层次，无淘汰。 操作系统实验平台的评测端基于分布式评测技术，可扩展性、可伸缩性强，可以在有限经费的情况下，灵活运用学校机房中的空闲计算机运算资源，将评测计算任务卸载至这些边缘设备上，从而应对多学生同时提交评测任务场景时的高频并发请求。在并发压力较低的时间段，也可以关闭部分评测终端，从而节省资源，提高资源利用率。相较于其他实验平台，本平台能利用更少的服务器主机资源，在不购置更

	多硬件设备的情况下，为学生和教师提供流畅而又功能强大的评测服务，从而更利于教学方案的实际落地。 与此同时，本套解决方案不仅可以运用在我校的操作系统课程教学中。在本方案于我校落地并达到稳定运行的状态后，我们也可以将其运用到与我校相同层次的学校的操作系统课程教学中去。本套方案涵盖了关于操作系统实验教学的方方面面，从技术文档，再到实验题目与实验指导书，再到评测平台与成绩系统，是一套成体系且考虑完备的解决方案，可以被直接应用各大高校的实际教学当中，满足绝大部分教学需求。同时，我们的解决方案有效地适应了不同学校的硬件资源差异、学生层次差异、教学水平差异，适应性强，可扩展性强，易于在各大高校中进行推广与实施。本方案应用到其他高校的过程中，可以通过授权费等形式形成盈利点，从而为本作品实现一定的经济效益。在此之上，由于本方案标准化程度较高，在应用到其他高校时，并不需要做过多的调整，很大程度地节省了部署实施时对接其他高校的人力成本。当我们的方案被各大高校广泛采用后，形成了一定的规模，我们方案部署所需的人力成本将会进一步减少，

	形成规模效应。 **参考文献：** 1.MIT操作系统开放实验， https://pdos.csail.mit.edu/6.828/2019/reference.html 2.清华大学操作系统实验指导书，uCore OS实验指导书和源码网址， https://chyyuu.gitbooks.io/ucore_os_docs/content/ 3.北京大学操作系统实验项目NachOS， https://github.com/wuhao9714/myNachos3.4

使用说明及该作品的技术特点和优势，提供该作品的适应范围及推广前景的技术性说明及市场分析和经济效益预测	**（一）使用说明：**1.学生实验流程： 查看实验题目与实验指导书利用虚拟机软件加载实验环境镜像->在所提供系统内核框架的基础上编写实验代码->利用编译工具链编译->得到可执行的系统内核->利用 QEMU 加载内核并模拟执行->利用 GDB 等工具调试，发现问题并修改代码利用工具，上传代码至在线评测系统->在线评测系统对上传来的代码进行编译，并用 QEMU 模拟加载编译得到的内核，执行测试用例，得出评测结果，并根据结果进行打分，评测结果和分数发送给成绩系统->成绩系统将成绩存储入库，并以界面的形式供学生访问查询，查看测试用例的通过情况。 2.教师监督教学情况流程： 打开成绩系统后台 查看不同学生的实验完成情况（测试用例通过情况及分数）->期末时导出平时成绩。 **（二）技术特点和优势：**解决了之前操作系统实验模拟算法使学生对操作系统原理理解不够深入的问题，通过在真正的操作系统内核进行实践，不仅提高了学生的实践动手能力，也提高了学生系统解决能力，此外分布式自动评测技术的引入方便了学生提交作业并及时查看实验结果并减轻了老师实验验机的负担。

	（三）作品适用范围：本作品适用于各大高校的操作系统实验使用。 **（四）推广前景：**本作品完成后先在本校试行，待作品成熟后，向其他学校推广。 **（五）市场分析和经济效益预测：**当下国内很多高校的操作系统实验没有专门的实验平台，相对于清华大学的 ucore 实验，本实验平台的题目难度较为适中，对普通高校学生而言，本实验平台的题目更容易上手，也更适合在普通高校的操作系统实验中推广使用。所以本作品的受众更广，更容易在各大高校中推广使用。操作系统实验平台的评测端基于分布式评测技术，可扩展性、可伸缩性强，可以在有限经费的情况下，灵活运用学校机房中的空闲计算机运算资源，将评测计算任务卸载至这些边缘设备上，从而应对多学生同时提交评测任务场景时的高频并发请求。在并发压力较低的时间段，也可以关闭部分评测终端，从而节省资源，提高资源利用率。相较于其他实验平台，本平台能利用更少的服务器主机资源，在不购置更多硬件设备的情况下，为学生和教师提供流畅而又功能强大的评测服务，从而更利于教学方案的实

	际落地。与此同时，本套解决方案不仅可以运用在我校的操作系统课程教学中。在本方案于我校落地并达到稳定运行的状态后，我们也可以将其运用到与我校相同层次的学校的操作系统课程教学中去。本套方案涵盖了关于操作系统实验教学的方方面面，从技术文档，再到实验题目与实验指导书，再到评测平台与成绩系统，是一套成体系且考虑完备的解决方案，可以被直接应用各大高校的实际教学当中，满足绝大部分教学需求。同时，我们的解决方案有效地适应了不同学校的硬件资源差异、学生层次差异、教学水平差异，适应性强，可扩展性强，易于在各大高校中进行推广与实施。本方案应用到其他高校的过程中，可以通过授权费等形式形成盈利点，从而为本作品实现一定的经济效益。在此之上，由于本方案标准化程度较高，在应用到其他高校时，并不需要做过多的调整，很大程度地节省了部署实施时对接其他高校的人力成本。当我们的方案被各大高校广泛采用后，形成了一定的规模，我们方案部署所需的人力成本将会进一步减少，形成规模效应。 因此，本作品的市场很广泛，且经济效益前景良好。

B. 当前国内外同类课题研究水平概述

国内外各高校在操作系统课程实验设计上都进行了改革，MIT 的 XV6，清华大学的 ucore、北京大学、以及北京航空航天大学等高校都改革了操作系统实验环节，让学生自己动手去设计一个操作系统，或者在部署好的虚拟环境上深入操作系统内核进行实践或实验。

MIT 的实验平台：6.S081 是美国麻省理工学院的操作系统课程实验，其课程参考资料较为全面，题目设计也十分巧妙，比较深入，比较贴合实际的工程实践。其所使用的 xv6 系统内核也是功能完整、资料丰富。但是这个实验的题目上手门槛较大，我校学生很难适应其题目难度。而且，其实验所考察的知识范围与我校教学大纲差异较大，其所涉及的很多知识点

在我校的课程中均没有深入讲解，学生直接上手的难度比较大，需要额外获取很多知识可能才能完成任务。此外，其资料虽然很完善，但是只有英文版，不利于我校学生学习。

清华大学的 ucore OS Labs：这是清华大学陈渝教授等主导的、清华大学操作系统实验用的 OS。实验代码开源，实验指导书、清华 Piazza 论坛也均可从外部访问，还有实验指导书的源码。上课学生主要参考的资料为学堂在线上的对应课程，并可以在实验楼平台中的对应课程中进行在线实验。上述 uCore 操作系统实验基于 qemu 虚拟化运行，使用 C 语言编写 CLI 程序进行交互，目标指令集架构为 x86 架构（32 位）。代码量（据实验指导书称）不多于五千行，参考了 xv6、OS161 和 Linux 内核等来设计出的操作系统，一般在 Linux 环境下进行实验。但是这些实验对学生基础要求比较高，对普通高校学生来说不适用。

华中科技大学的 OS 课程包含三个实验，均涉及系统 API，原理性的内容也有，但占比相对较低。报告中可见，实验涉及了大量进程间通信内容，也有关于线程的调用，但较少；但让学生弄明白线程和进程 是有必要的。从上下文来看，教职员工似乎没有提供任何可供参照的代码。使用的应该是 C 语言。 实验一要求学生写两个线程的程序，一个负责加一，一个负责输出，换言之，一个读一个写。很显然，这是想要学生了解多线程中出现的幻影读等脏数据现象，但原作者直接写了加信号量版本，而

且实验指导书似乎确实对此缺少要求。考虑到程序并不复杂，学生很容易存在互相“借鉴”的现象， 若学生都这么写程序，这个实验很可能无法达到预期的教学效果。笔者认为，可以考虑对这一点进 一步完善，给出已经写好的、没有做好同步的代码，要求学生进行修改，并比对修改前后的运行结 果差异。当然，这并不能根除学生偷懒的现象，但至少能让一部分原本思路不清晰的学生理解线程 同步操作的意义，理解并发程序中进行合理的同步操作的必要性。 实验二要求配合共享内存，实现用四个进程来传递文件内容。很显然，有些学生不怎么了解 C 语言的文件操作，绕过了这个要求，而且很可能是故意的。我们在设计实验时最好是不要有完全交 给学生自己写的题目，而是一律采用主体框架混合学生代码的形式。 实验三没有什么需要深究的内容，纯粹是 API 调用而已。坦诚地讲，从这份报告中我们也能受到启发，那就是别给学生太 多自由，但也应当简化掉一些 API 有关内容，对 API 的涉及最好是蜻蜓点水一般，以原理为重点。 如果 API 占的戏份太多，学生就可能在 API 上花费太多精力，而在真正的重点——程序逻辑上面， 采用一种“糊弄糊弄得了”的心态。此外，书写实验指导书时，应当以尽可能多的细节为目标，以防有学生阴差阳错地或者故意地忽略掉了教学人员真正想要着重强调的东西。

NachOS 是不少高校在使用的一个模拟 OS，其有 java、c 等多种语言

版本，使用起来比较灵活。它在实现上重逻辑轻底层，这使得它让学生更专注于算法本身，但也难以移植到硬件结构上。这使得学生在实现算法时不受到一些现实中的硬件限制，可能会降低对底层原理的理解体会效果。在我校教学实践总结出的经验中，我们也看到，熟悉算法的学生会过于执着于时间复杂度而忽略了工程上的常数和可维护性，对于我校这类情况而言，我们也希望学生能具有对工程实践的良好理解，并借此机会培养查阅文档写程序的良好习惯。