

2021全国大学生计算机系统能力大赛

操作系统设计赛

技术方案说明

王雷 陈渝 萧络元

2020年12月26日

钻石赞助:

麒麟软件

金牌赞助:

翼辉科技 蚂蚁集团

银牌赞助:

国科环宇 360 匿名捐赠者

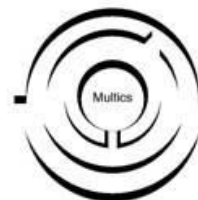
铜牌赞助:

OpenAnolis社区 中科创达 测测星座 全志科技 龙芯中科 元心科技 睿赛德rtthread

主办单位: 全国高等学校计算机教育研究会 系统能力培养研究专家组 系统能力培养研究项目示范高校

承办单位: 鹏城实验室 哈工大深圳分校 中科院计算所

os.educg.net github.com/oscomp



目标



• 开放&开源

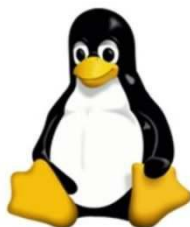
- 所有参赛项目的源代码和文档采用开源协议开源
- 欢迎多家公司赞助&参加 (做导师, 评委, 技术支持)
- 鼓励国内各个层次的高校的OS教师和学生参加 (参赛, 导师, 评委, 技术支持)



目标

- 可继承性的迭代发展

- 每届的成果能够继承并发展，通过开源方式推动学生可站在前人基础上进一步发展
- 鼓励合作型竞争，即鼓励参赛队在比赛过程中能够学习其他队的先进经验，并进行一定程度的合作

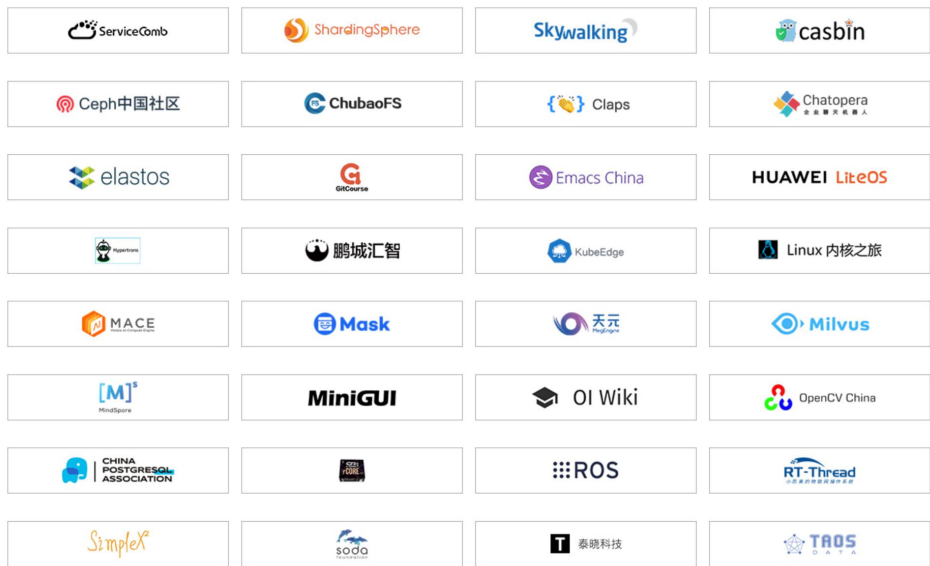
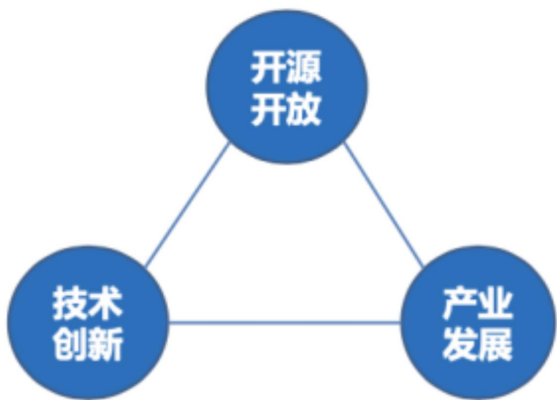
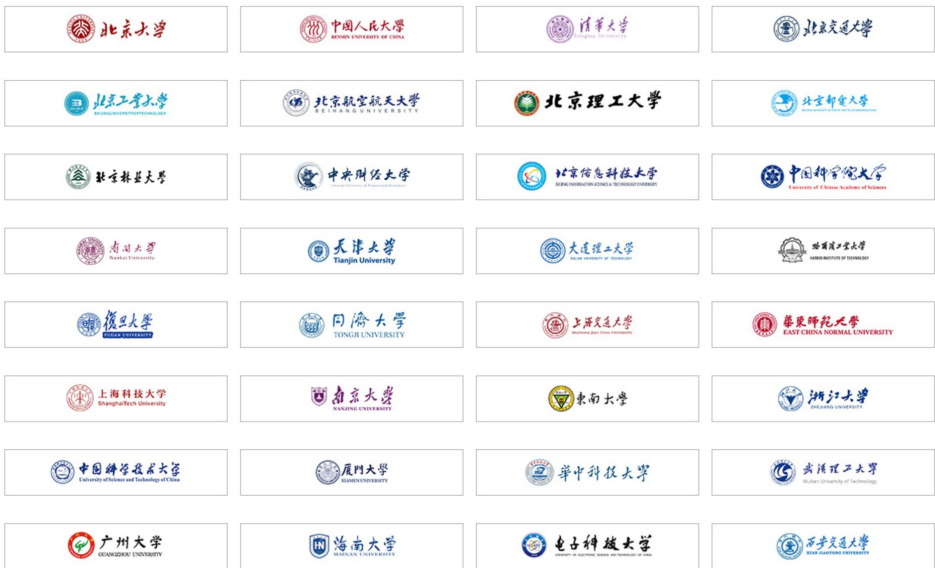


UNIX[®]
Multics

目标

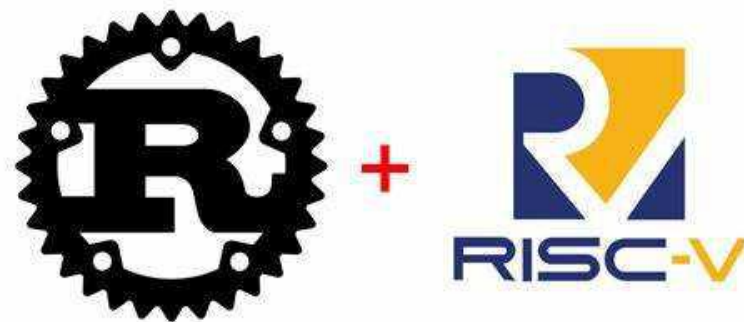
• 探索创新

- 不限定语言/架构，可以灵活组合的OS kernel设计
- 提供两个赛道，发挥学生，企业和教师的创造性
- 鼓励OS企业或研究单位讲解交流推广合作最新研究成果（要求开源）



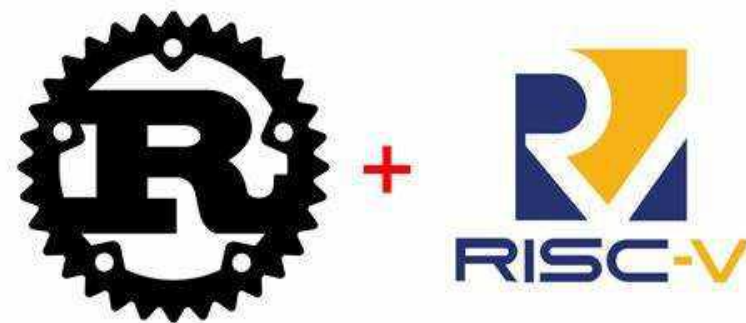
参赛队任务概览

- 比赛分成“OS功能设计”和“OS内核实现”两个类别
- “OS功能设计”比赛
 - 题目来源：与OS相关，来源于企业/学校/个人，内容比较灵活
 - 包括：课程实验，工具，编程语言，系统软件，软硬件协同设计等



参赛队任务概览

- 比赛分成“OS功能设计”和“OS内核实现”两个类别
- “OS功能设计”比赛
 - 比赛内容可帮助高校OS教学与实验，以及让学生了解企业的技术
 - 最后的评比结果基于全体评审专家的综合打分（主观性较大）



参赛队任务概览

- 比赛分成“OS功能设计”和“OS内核实现”两个类别
- “OS内核实现”比赛
 - 在规定硬件上实现规定系统调用的OS Kernel
 - 比赛内容推动学生和老师了解和掌握OS Kernel设计与实现
 - 最后的评比结果基于评测结果和评审专家的综合打分（客观性较大）

POSIX

Your DIY OS Kernel



参赛队任务概览

- 比赛分成“OS功能设计”和“OS内核实现”两个类别
- 比赛过程分初赛和决赛两个阶段
- 赛事流程

• 交流 -----> 学习 -----> 创造
• (1月~3月) (4月~6月) (7月~8月)



参赛队任务概览

- 比赛分成“OS功能设计”和“OS内核实现”两个类别
- 比赛过程分初赛和决赛两个阶段
- 赛事流程（报名阶段：线上）
 - 2021年1月1日~2021年3月31日
 - 报名：一个学校可以有4个队，OS内核实现最多2个，OS功能设计最多2个，每队最多3人（同一学校的本科生），一个学生报一个队
 - OS内核实现赛：发布指定实验平台，发布初赛具体评测技术指标
 - OS功能设计赛：发布参赛题目，给出初赛评审参考意见
 - 定期组织技术交流

参赛队任务概览

- 比赛分成“OS功能设计”和“OS内核实现”两个类别
- 比赛过程分初赛和决赛两个阶段
- 赛事流程（初赛阶段：线上）
 - 2021年4月1日~2021年5月31日
 - 参赛学生开始完成初赛题目
 - 初赛完成过程中要求源码与开发过程都开源，提供访问网址
 - 定期组织技术交流
 - 2021年6月1日~2021年6月20日
 - 初赛作品评审，公布入围全国总决赛的参赛队名单

参赛队任务概览

- 比赛分成“OS功能设计”和“OS内核实现”两个类别
- 比赛过程分初赛和决赛两个阶段
- 赛事流程（决赛阶段：线上）
 - 2021年6月20日
 - 对于OS内核实现赛，发布参赛题目，公布决赛的具体评测技术指标
 - 2021年6月21日~2021年7月20日
 - OS内核实现参赛队公开现阶段源代码，评测组给决赛第一次提交的成绩或排名；定期组织技术交流
 - 2021年7月21日~2021年8月18日
 - OS内核实现参赛队公开现阶段源代码，评测组给决赛第二次提交的成绩或排名；定期组织技术交流

参赛队任务概览

- 比赛分成“OS功能设计”和“OS内核实现”两个类别
- 比赛过程分初赛和决赛两个阶段
- 赛事流程（决赛阶段：线下）
 - 2021年8月21日
 - 对于OS内核实现赛，参赛队现场完成指定题目；对于OS功能设计赛，参赛队现场答辩
 - 2021年8月22日
 - 参赛队现场答辩&颁奖典礼

参赛队任务概览

- 参赛作品的源代码遵循开源协议

- GPL-3协议\Apache协议\BSD协议\木兰协议等，参赛作品的文档遵循开源协议如GNU FDL协议\CC协议等。

- 参赛队应自觉遵守知识产权和开源协议的法规 and 规定

- 不得侵犯他人的知识产权或其他权益。如造成不良后果，相关法律责任由参赛队自行承担，本竞赛的主办、承办和协办方均不负任何法律责任。

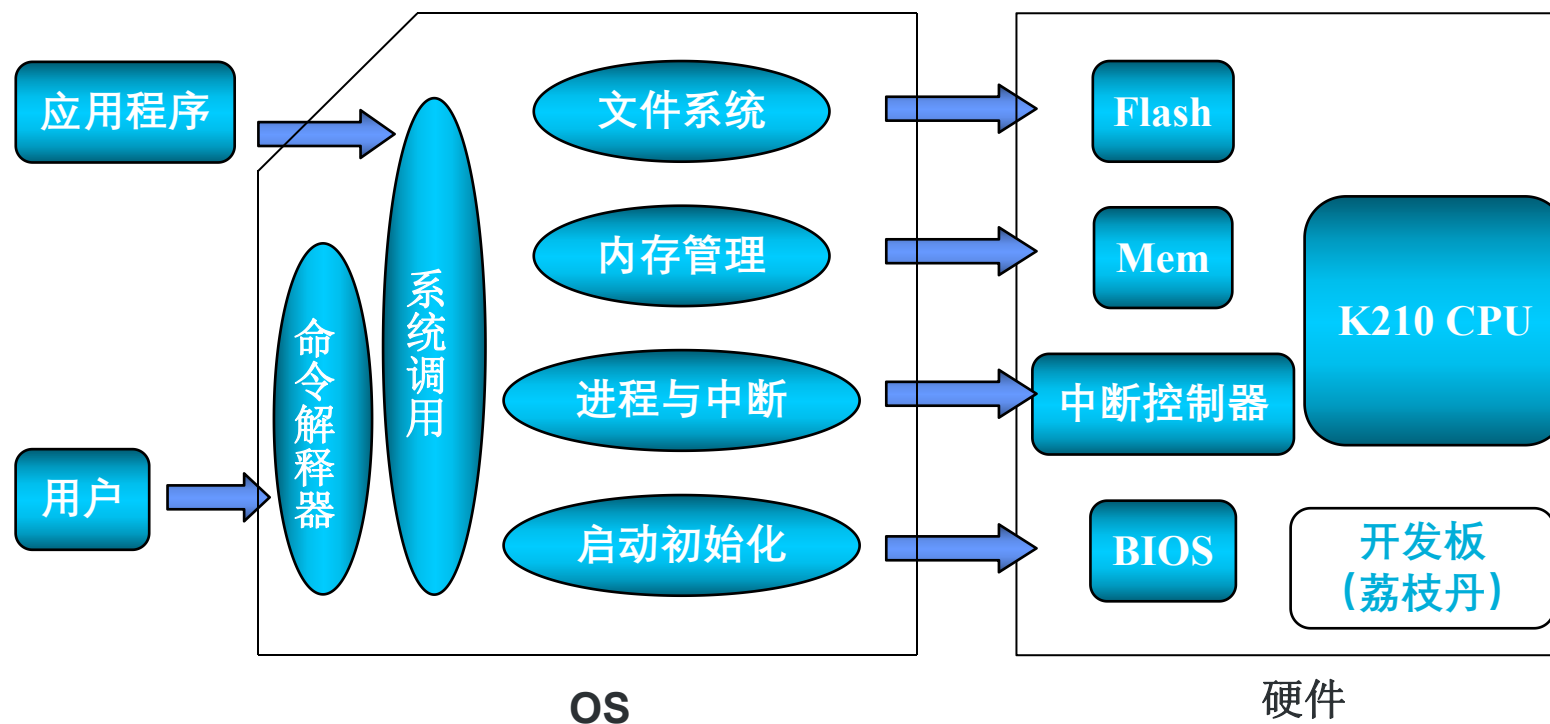
- 参赛队应保证学术诚信

- 如收到实名举报或评审中发现可能的代码、文档或技术等方面的恶意抄袭等学术不端行为，将由比赛专家组授权的学术诚信委员会组织匿名专家进行鉴定和判断，一经确认为学术不端行为，将即时取消参赛资格。对于已经获得奖项的参赛队，通报所在学校，并取消和追回已获的的证书和奖金。

OS内核实现赛的评测说明

- 评测过程

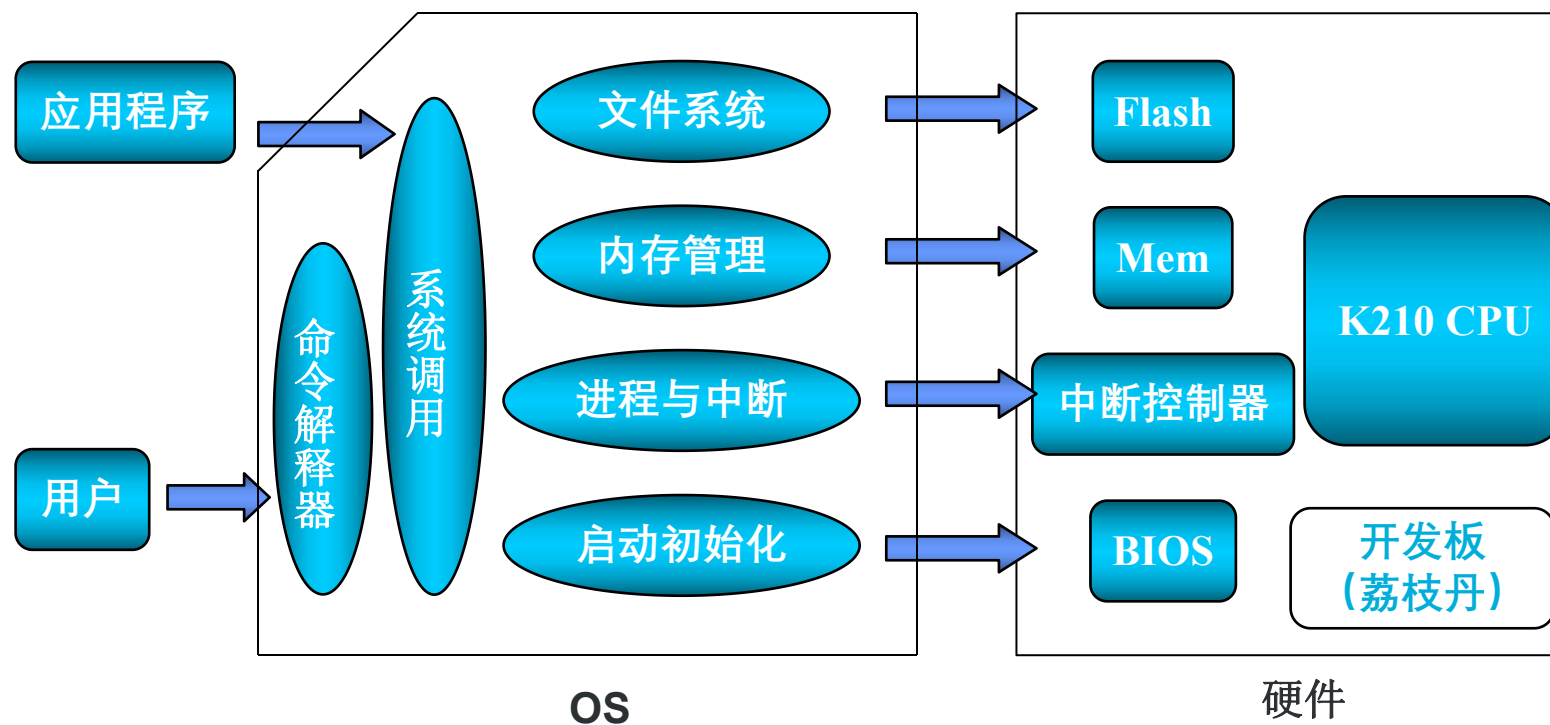
- 参赛队实现的OS Kernel能访问有规定的文件系统上的应用程序和数据。基准测试程序放在规定的文件系统上。参赛队在完成规定的文件系统情况下，可实现新的文件系统，提高测试程序的执行性能。



OS内核实现赛的评测说明

- 评测过程

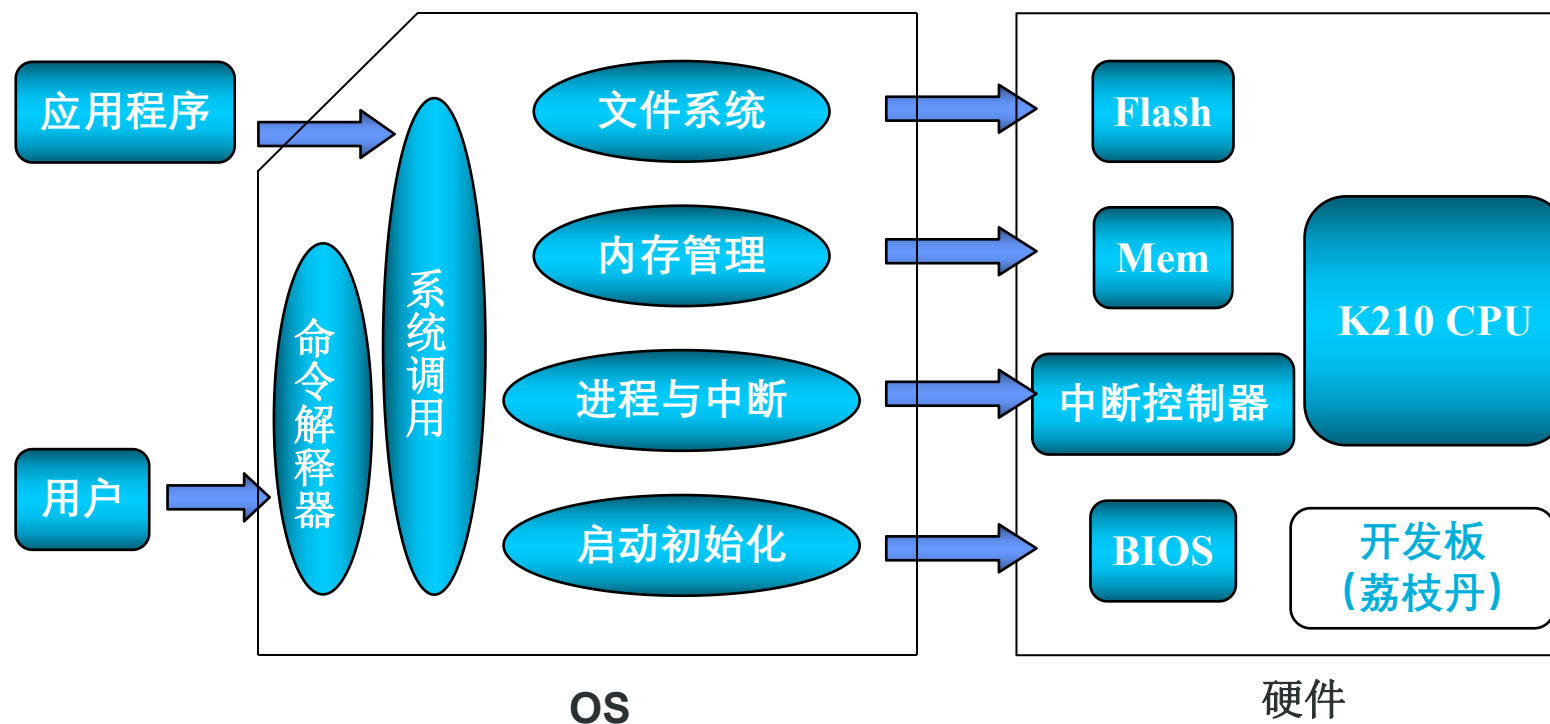
- 功能与性能测试是指在大赛规定的物理硬件平台上，分析每个基准测试程序在参赛队所开发的操作系统上的运行情况。每个基准测试程序性能指标没通过或未能正确运行（二者都表现为程序未运行通过）计0分,运行通过计1分。参赛队的最终功能与性能测试成绩为通过的基准测试程序个数。



OS内核实现赛的评测说明

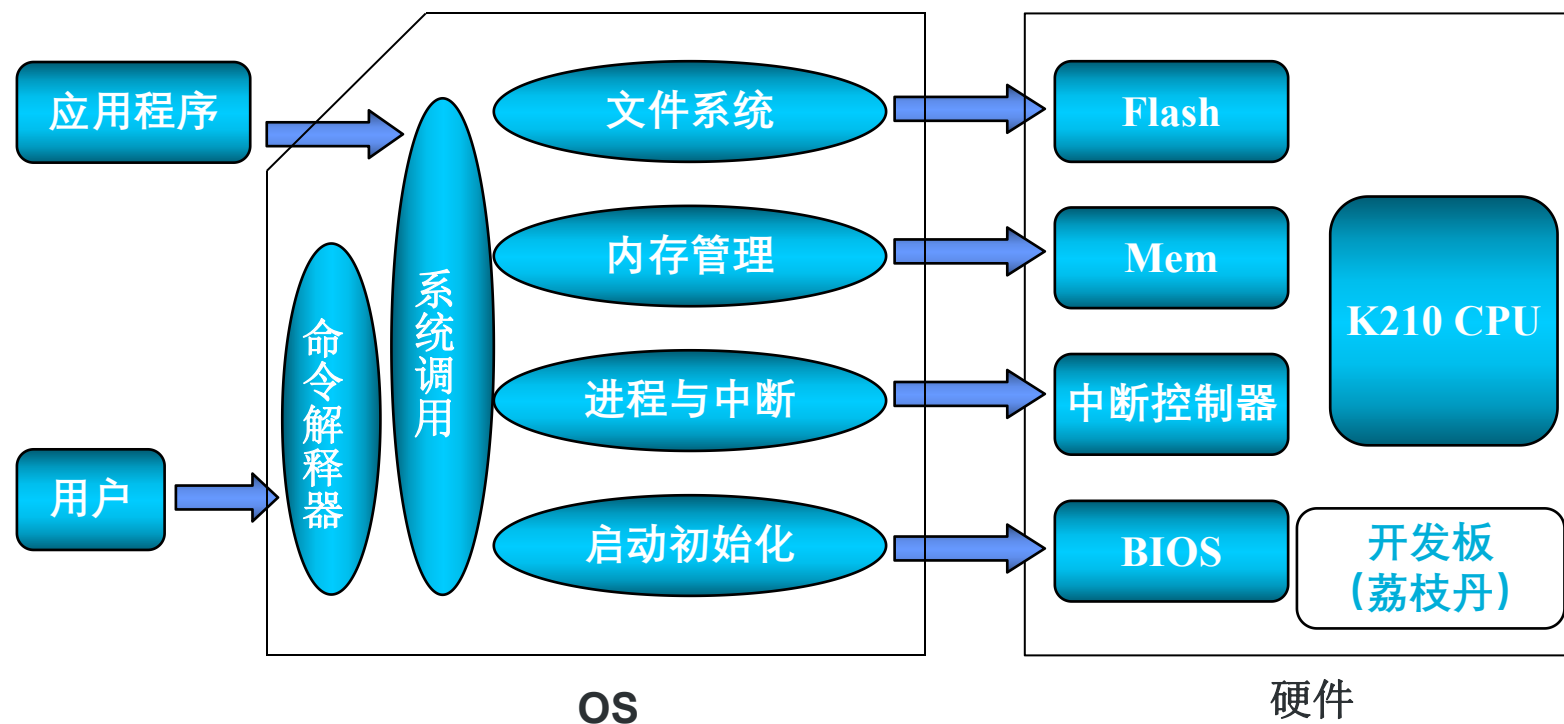
- 评测过程

- 对于漏洞分析工具分析出的Bug，需要能及时修复或直接指出是误报。对于没有修复或指出是误报的Bug的个数将形成负分，计入总成绩。

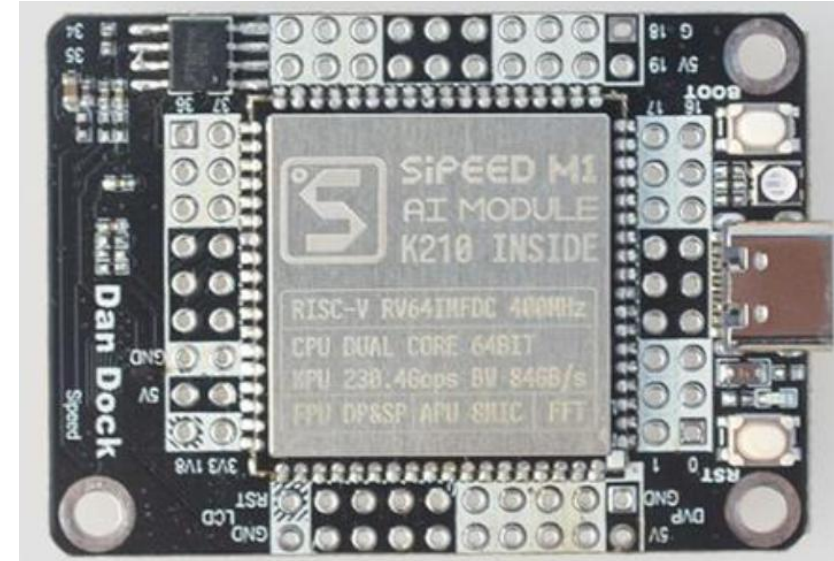


参赛操作系统期望功能

- 启动和系统初始化
- 内存管理
- 进程管理和中断异常机制
- 系统调用
- 文件系统 (SimpleFS or Other better FS)
- 命令解释程序



	MAIX 系列开发板
主控芯片	K210
芯片内核	双核RISC-V 64bit IMAFDC
主频	标称400M，可超600M
内存	6MB 通用内存+2MB AI内存
Flash	16MB
mpy支持	支持，MaixPy开源项目



评测机：

Sipeed M1(荔枝丹)开发板，处理器Kendryte K210

核心单元：

双核处理器，带独立FPU
CPU位宽：64位
8M 片上SRAM，
400M 可调标称频率，
支持乘法、除法和平方根运算 双精度FPU

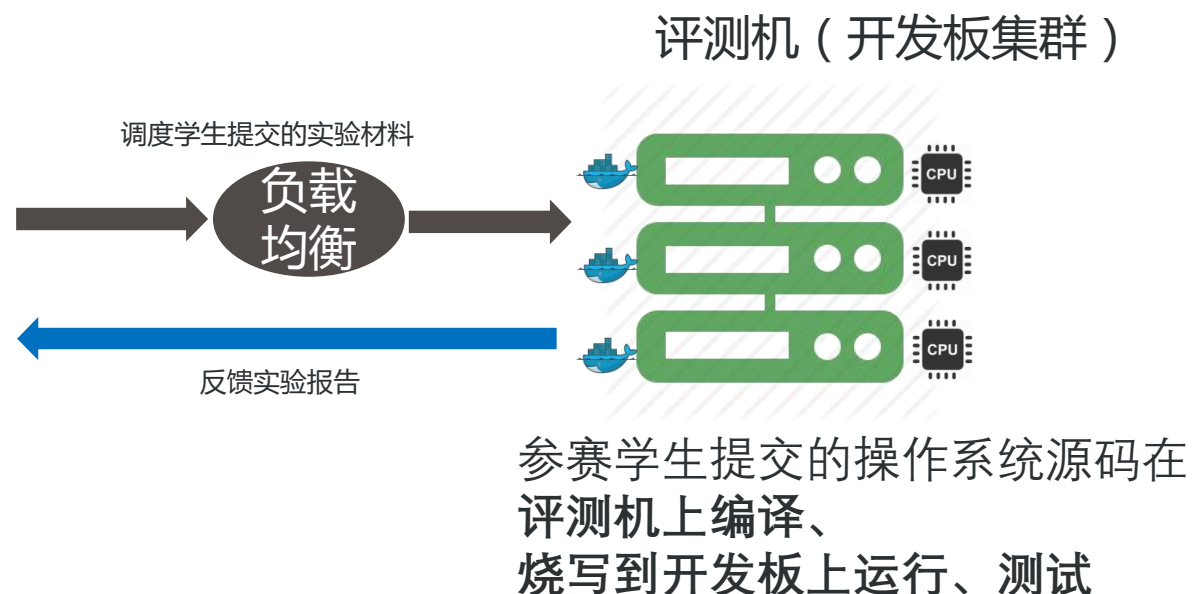
软件环境：

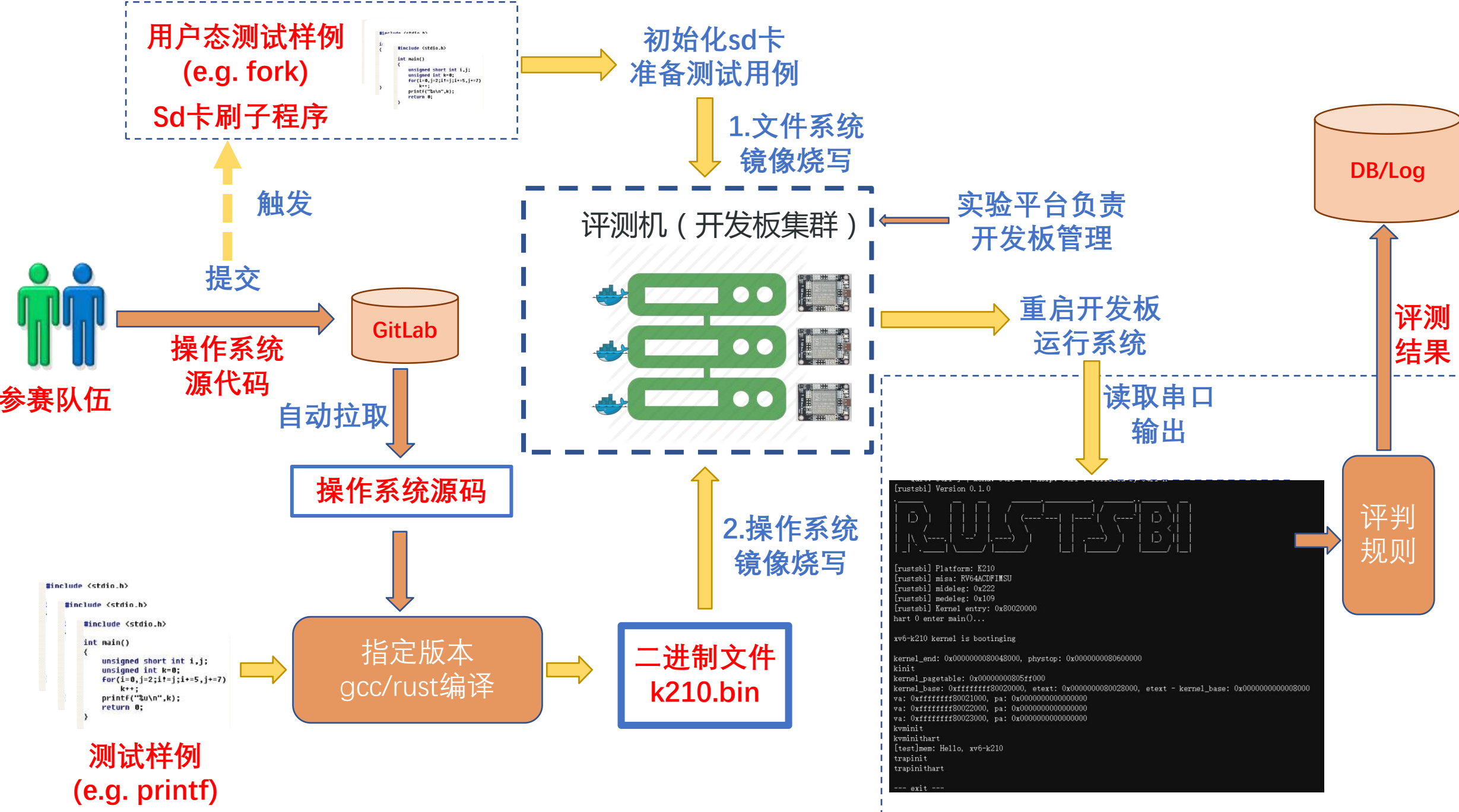
操作系统：ubuntu18.04

编译工具：

RISC-V Toolchain、
kendryte-standalone-sdk

烧写工具：RustSBI、kflash





自动评测实验过程

- 克隆

- 通过学生提交的http仓库链接克隆仓库
- 拉取到/coursegrader/submit文件夹下

- 初始化sd卡

- 将刷子程序与准备好的文件系统镜像（测试用例）打包并烧写到开发板上
- 刷子程序从指定位置将文件系统镜像复制到sd卡上，完成sd卡的刷写过程

- 编译

- Makefile中规定make k210命令
- 评测机搭载RISC-V交叉编译工具链([kendryte-gnu-toolchain](#))
- 编译输出为/coursegrader/submit/target/k210.bin文件

- 烧写

- ssh连接到搭载有开发板的Docker容器。
- 将编译好的 .bin 文件上传到容器中
- 容器执行脚本 ssh_run.py 使用kflash烧写.bin文件

-

```
kflash -b 3000000 -B dan /cg/k210.bin -p {usb_port} > /cg/k210_flash_out.txt
```

自动评测实验过程

- 读取串口信息

```
ss = serial.Serial(usb_port, 115200, # usb_port (default) : /dev/ttyUSB0
                    timeout=2,
                    parity=serial.PARITY_NONE,
                    stopbits=serial.STOPBITS_ONE,
                    bytesize=serial.EIGHTBITS)
```

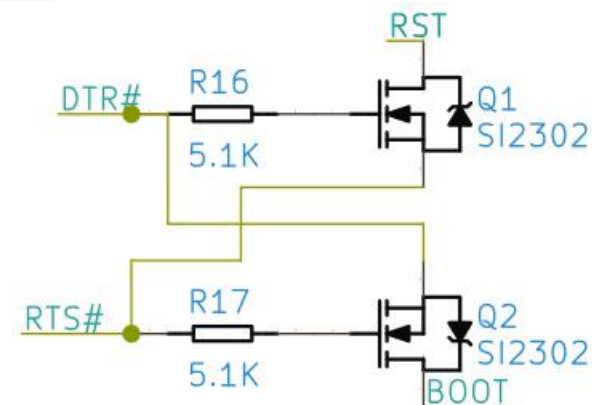
- 逐行读取串口输出并解码，结果保存于k210_serial_out.txt文件中。

- 输出结果评测

- 根据评判规则（boot过程输出、测试点等）对操作系统标准输出进行评测

- 异常处理

- 串口传输、读取过程中出现bit丢失
—— 捕捉解码异常
- 操作系统运行结果不一致（待定）



评测机拟支持环境

开发语言、编译器以及库依赖

Linux

(麒麟, 统信, OpenEuler, Ubuntu 18.04)

gcc

*riscv64-unknown-elf-gcc (SiFive GCC 8.3.0-
2020.04.0) 8.3.0*

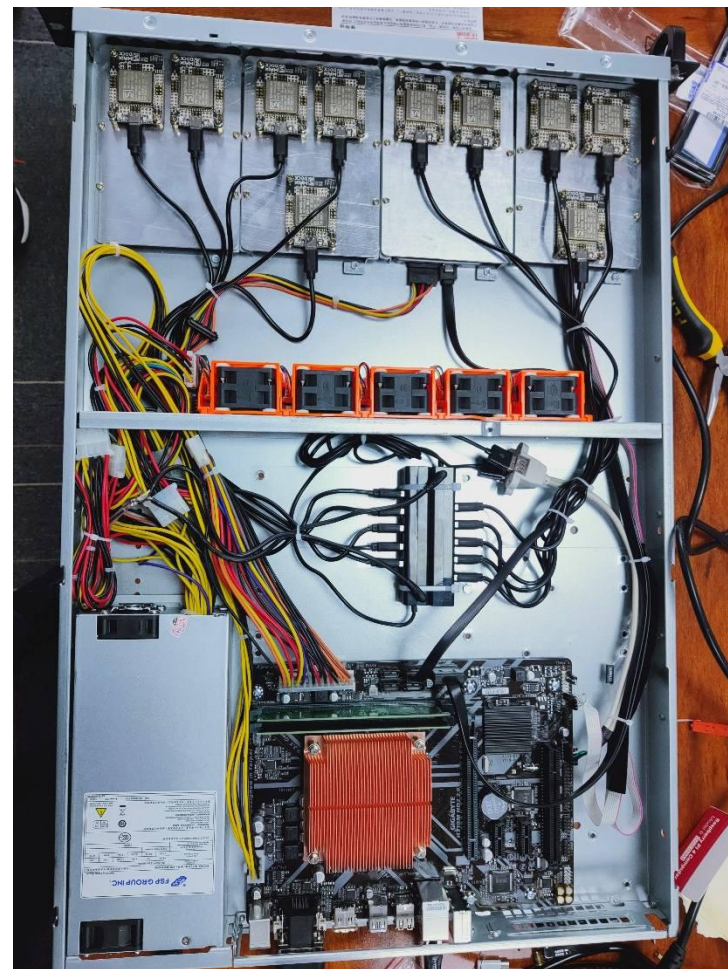
——
*[https://github.com/loboris/ktool/tree/master/kendryte-
toolchain/bin](https://github.com/loboris/ktool/tree/master/kendryte-toolchain/bin)*

Rust

nightly-2020-06-27

—— 目前支持: rCore-Tutorial所依赖的包
烧写环境(kflash or ktool)

Python 3.7及以上



测试用例示例

系统调用测试

规定系统调用号与参数——以rCore-Tutorial为例：

简单测试用例的系统调用库示例

```
// syscall
int64_t syscall_63(); // SYSCALL 63 @ const SYSCALL_READ: usize = 63;
int64_t syscall_64(); // SYSCALL 64 @ const SYSCALL_WRITE: usize = 64;
int64_t syscall_93(); // SYSCALL 93 @ const SYSCALL_EXIT: usize = 93;
int64_t syscall_101(); // SYSCALL 101 @ const SYSCALL_SLEEP: usize = 101;
int64_t syscall_124(); // SYSCALL 124 @ const SYSCALL_YIELD: usize = 124;
int64_t syscall_129(); // SYSCALL 129 @ const SYSCALL_KILL: usize = 129;
int64_t syscall_169(); // SYSCALL 169 @ const SYSCALL_GET_TIME: usize = 169;
int64_t syscall_172(); // SYSCALL 172 @ const SYSCALL_GETPID: usize = 172;
int64_t syscall_220(); // SYSCALL 220 @ const SYSCALL_FORK: usize = 220;
int64_t syscall_221(); // SYSCALL 221 @ const SYSCALL_EXEC: usize = 221;
int64_t syscall_260(); // SYSCALL 260 @ const SYSCALL_WAIT: usize = 260;
```

```
void putc(char c) {
    char buf[1];
    buf[0] = c;
    syscall_64(1, buf, 1);
}

void puts(char *str) {
    int length = 0;
    char *q = str;
    while (*q != 0)
    {
        length++;
        q++;
    }
    syscall_64(1, str, length);
}

int fork(){
    return syscall_220(0,0,0,0,0);
}

int getpid(){
    return syscall_172();
}

int gettimeofday(){
    return syscall_169();
}
```

```
int getpid(){
    return syscall_172();
}

int gettimeofday(){
    return syscall_169();
}

void sleep(int ticks){
    syscall_101(ticks);
}

void yield(){
    syscall_124();
}

int kill(int pid){
    return syscall_129(pid);
}
```


测试用例示例

系统调用测试

规定系统调用号与参数

简单测试用例设计示例：

—— 该测试用例主要测试SYS_WRITE与SYS_FORK两个系统调用

测试代码

```
#include "stdtest.h"
You, 2 weeks ago • initial
void test_fork()
{
    puts("\n---[test_fork] BEGIN---\n");
    int fork_ret = fork();
    if (fork_ret == -1)
    {
        puts("fork error!\n");
    }
    if (fork_ret == 0)
    {
        puts("[fork]");
        puts("this is child\n");
        exit();
    }
    else
    {
        /* parent */
        puts("[fork]");
        puts("this is parent\n");
    } /* TEST_RETURN */
    puts("\n---[test_fork] END\n---");
}
```

系统输出(rCore-Tutorial)

```
mod fs initialized
mod interrupt initialized
mod interrupt initialized
Hello rCore hartid = 0, dtp_pa = PhysicalAddress(0x2333333366666666)
Hello rCore hartid = 1, dtp_pa = PhysicalAddress(0x2333333366666666)
=====
|--BEGIN TEST--|
=====

---[test_fork] BEGIN---

[fork]this is child
[fork]this is parent

---[test_fork] END---
=====
|---END TEST---|
=====
```

平台反馈结果(测试中)

13163}, "HTML": "enable", "comment": "				
name	verdict	score	time	comments
TestCase 1	Accept	100.0	13163	[test1] correct!
				[test2] correct!
				[test3] correct!
", "detail": "				
TestCase 1				
output		answer		
{ 'verdict': 'Accept' }				
"				

OS比赛官网: os.educg.net

学生自行注册账号报名参赛



全国大学生计算机系统能力大赛
编译系统赛+操作系统赛 官方指定大赛平台

主办单位: 全国高等学校计算机教育研究会 系统能力培养研究项目示范高校
协办单位: 华为技术有限公司 机械工业出版社华章分社

2021全国大学生计算机系统能力大赛-操作系统设计赛 参赛人数 1

初赛 决赛 结束
11/08 21:09 - 12/31 21:05 11/11 10:30 - 12/06 11:35 08/26 12:00

报名未开始

介绍 排行榜

大赛介绍

全国大学生计算机系统能力大赛(以下简称“大赛”)是由教育部高等学校计算机类专业教学指导委员会和系统能力培养研究专家组共同发起,以学科竞赛推动专业建设和计算机领域创新人才培养体系改革、培育我国高端芯片及核心系统的技术突破与产业化后备人才为目标,面向高校大学生举办的全国性大赛。操作系统设计赛要求各参赛队综合运用各种知识(包括但不限于编译技术、操作系统、计算机体系结构等),构思并实现一个操作系统,以展示面向特定目标平台的操作系统设计能力。

编译系统设计赛要求各参赛队综合运用各种知识(包括但不限于编译技术、操作系统、计算机体系结构等),构思并实现一个综合性的编译系统,以展示面向特定目标平台的编译器构造与编译优化的能力。

大赛官网: os.educg.net

大赛章程: 待定

技术方案: 待定

主办单位: 全国高等学校计算机教育研究会、清华大学、北京大学、北京航空航天大学、国防科学技术大学、南京大学、上海交通大学、浙江大学、中国科学技术大学

协办单位: 华为技术有限公司、机械工业出版社华章分社

OS比赛题目列表: <https://github.com/oscomp/>

OS比赛评测入口



比赛报名



组建团队



提交结果



评分排名



入围决赛



决赛答辩

course.educg.net/sv2/indexexp/contest/index.jsp

希冀

比赛

OnlineJudge

GitLab

教学系统

登录 注册 教师入口

所有比赛

进行中

未开始

已结束

2021全国大学生计算机系统能力大赛-编译系统设计赛

2021编译系统设计赛

未开始

2021全国大学生计算机系统能力大赛-编译系统设计赛

2021编译系统设计赛

2021编译系统设计赛

全国大学生计算机系统能力大赛

编译系统设计赛+操作系统设计赛 官方指定大赛平台

主办单位: 教育部高等学校计算机类专业教学指导委员会和系统能力培养研究专家组

承办单位: 希冀教育

大赛官网: compiler.educg.net

① 竞赛时间 2021/01/01 - 2021/08/16

📅 报名截止时间 2021-03-01 09:33:00

👤 参赛人数 0

👁 浏览数 223

2021全国大学生计算机系统能力大赛-操作系统设计赛

2021操作系统设计赛

未开始

2021全国大学生计算机系统能力大赛-操作系统设计赛

2021操作系统设计赛

2021操作系统设计赛

全国大学生计算机系统能力大赛

编译系统设计赛+操作系统设计赛 官方指定大赛平台

主办单位: 教育部高等学校计算机类专业教学指导委员会和系统能力培养研究专家组

承办单位: 希冀教育

大赛官网: os.educg.net

① 竞赛时间 2021/01/01 - 2021/08/26

📅 报名截止时间 2021-03-01 10:00:00

👤 参赛人数 1

👁 浏览数 119

2020全国大学生计算机系统能力大赛-编译系统设计赛-赛后通道

2020编译系统赛-赛后评测

进行中

2020全国大学生计算机系统能力大赛-编译系统设计赛-赛后通道

2020编译系统赛-赛后评测

2020编译系统赛-赛后评测

2020年全国大学生计算机系统能力大赛

编译系统设计赛

主办单位: 教育部高等学校计算机类专业教学指导委员会和系统能力培养研究专家组

承办单位: 希冀教育

大赛官网: compiler.educg.net

① 竞赛时间 2020/11/04 - 2021/03/31

📅 报名截止时间 2021-03-01 11:11:00

👤 参赛人数 82

👁 浏览数 180

C语言程序设计 互联网20-1,2班 第4次实验

C语言程序设计 互联网20-1,2班 第4次实验

进行中

C语言程序设计 互联网20-1,2班 第4次实验

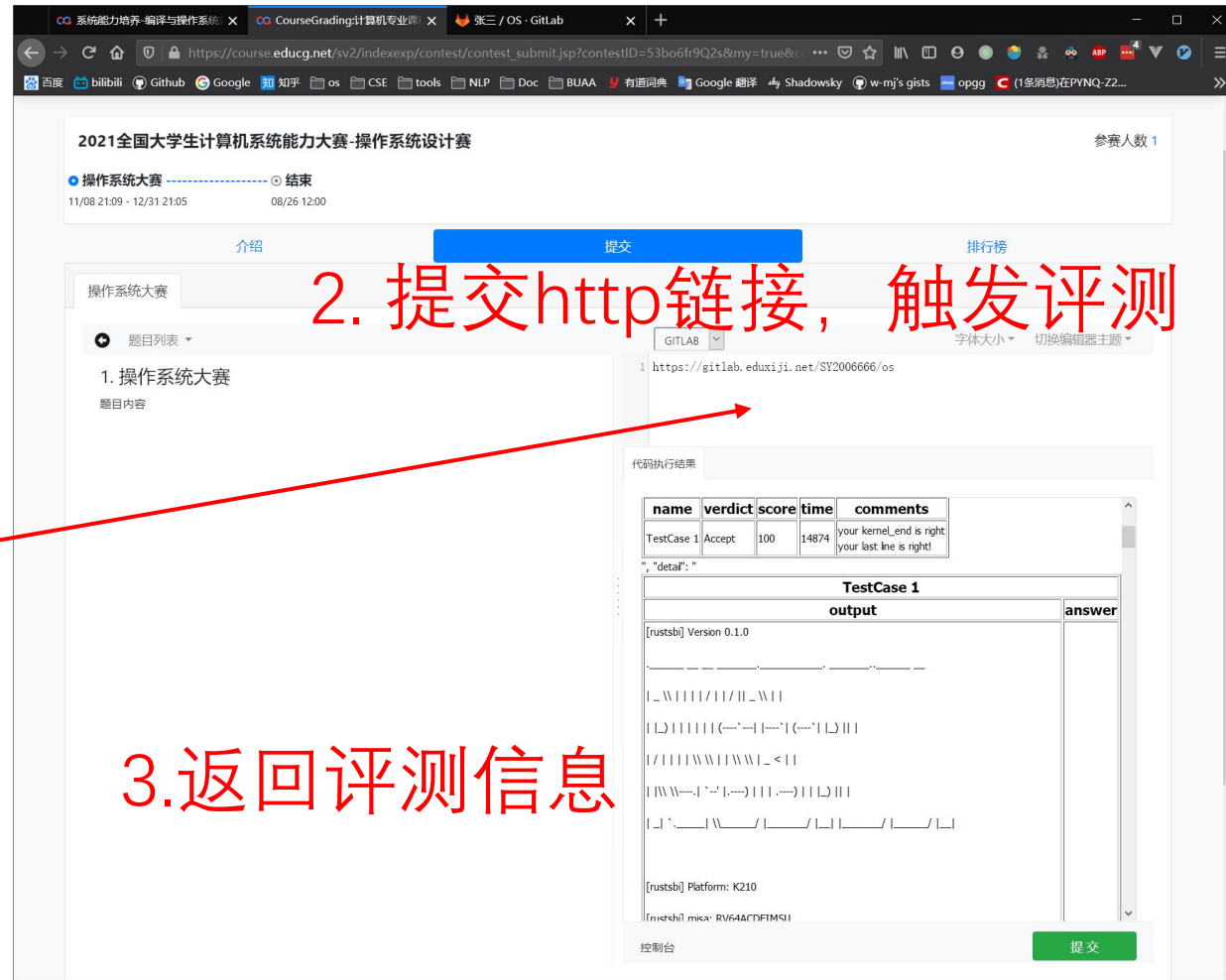
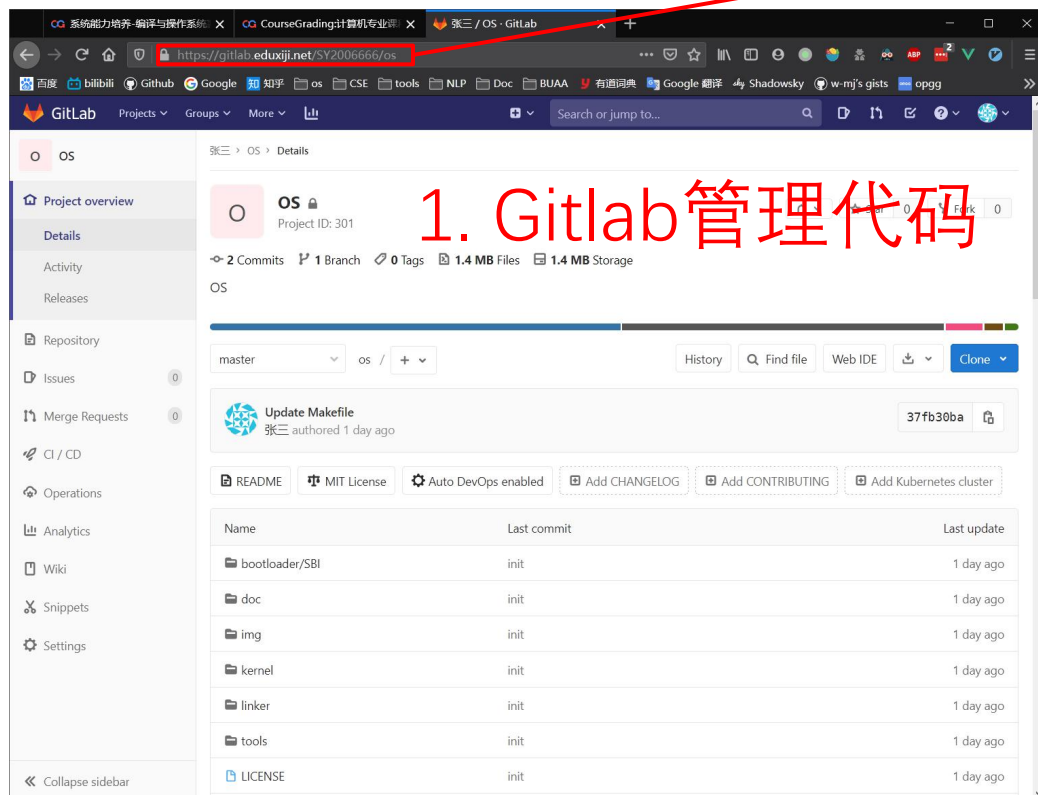
C语言程序设计 互联网20-1,2班 第4次实验

C语言程序设计 互联网20-1,2班 第4次实验

希冀

OS比赛结果提交

- 在本地完成竞赛要求的操作系统开发
 - 需要对Makefile格式做相应规定 (待定)
- 提交源代码到gitlab仓库



- 将gitlab项目链接复制到cg平台
- 点击提交, 平台自动clone代码仓库, 编译、烧写、评测后返回评测信息

后续需要准备的文档

- ⑩ K210 SDK: <https://canaan-creative.com/wp-content/uploads/2020/03/kendryte-standalone-sdk-0.5.6.zip>
- ⑩ RISC-V: <https://riscv.org/specifications/>
- ⑩ Linux系统调用规范。
- ⑩ 测试用例范例。
- ⑩ Simple FS/FAT32文件系统规范。（待定）
- ⑩ 竞赛平台（代码托管平台、竞赛测试系统）使用文档。
- ⑩ 操作系统基本框架与Makefile格式要求。（待定）
- ⑩ 参赛队远程调试及本地调试指南。（待定）

系统调用约定

- ⑩ 系统调用是一种程序进入内核执行任务的方式。
- ⑩ 触发系统调用需要特殊的指令和存放参数的寄存器。
- ⑩ 例如i386架构的int指令，RISCV架构的ecall指令，然后通过特定寄存器传递参数

Arch/ABI	Instruction	System call #	Ret val	Ret val2	Error	Notes
alpha	callsys	v0	v0	a4	a3	1, 6
arc	trap0	r8	r0	-	-	
arm/OABI	swi NR	-	r0	-	-	2
arm/EABI	swi 0x0	r7	r0	r1	-	
arm64	svc #0	w8	x0	x1	-	
blackfin	excpt 0x0	P0	R0	-	-	
i386	int \$0x80	eax	eax	edx	-	
ia64	break 0x100000	r15	r8	r9	r10	1, 6
m68k	trap #0	d0	d0	-	-	
microblaze	brki r14,8	r12	r3	-	-	
mips	syscall	v0	v0	v1	a3	1, 6
nios2	trap	r2	r2	-	r7	
parisc	ble 0x100(%sr2, %r0)	r20	r28	-	-	
powerpc	sc	r0	r3	-	r0	1
powerpc64	sc	r0	r3	-	cr0.S0	1
riscv	ecall	a7	a0	a1	-	
s390	svc 0	r1	r2	r3	-	3
s390x	svc 0	r1	r2	r3	-	3
superh	trap #0x17	r3	r0	r1	-	4, 6
sparc/32	t 0x10	g1	o0	o1	psr/csr	1, 6
sparc/64	t 0x6d	g1	o0	o1	psr/csr	1, 6
tile	swint1	R10	R00	-	R01	1
x86-64	syscall	rax	rax	rdx	-	5
x32	syscall	rax	rax	rdx	-	5
xtensa	syscall	a2	a2	-	-	

系统调用约定

系统调用实现函数以riscv为例：

寄存器**a7**传递系统调用号，**a0-a5**作为其他参数；

syscall:

mv a7, a0

mv a0, a1

mv a1, a2

mv a2, a3

mv a3, a4

mv a4, a5

mv a5, a6

ecall

ret

Arch/ABI	arg1	arg2	arg3	arg4	arg5	arg6	arg7	Notes
alpha	a0	a1	a2	a3	a4	a5	-	
arc	r0	r1	r2	r3	r4	r5	-	
arm/OABI	r0	r1	r2	r3	r4	r5	r6	
arm/EABI	r0	r1	r2	r3	r4	r5	r6	
arm64	x0	x1	x2	x3	x4	x5	-	
blackfin	R0	R1	R2	R3	R4	R5	-	
i386	ebx	ecx	edx	esi	edi	ebp	-	
ia64	out0	out1	out2	out3	out4	out5	-	
m68k	d1	d2	d3	d4	d5	a0	-	
microblaze	r5	r6	r7	r8	r9	r10	-	
mips/o32	a0	a1	a2	a3	-	-	-	1
mips/n32,64	a0	a1	a2	a3	a4	a5	-	
nios2	r4	r5	r6	r7	r8	r9	-	
parisc	r26	r25	r24	r23	r22	r21	-	
powerpc	r3	r4	r5	r6	r7	r8	r9	
powerpc64	r3	r4	r5	r6	r7	r8	-	
riscv	a0	a1	a2	a3	a4	a5	-	
s390	r2	r3	r4	r5	r6	r7	-	
s390x	r2	r3	r4	r5	r6	r7	-	
superh	r4	r5	r6	r7	r0	r1	r2	
sparc/32	o0	o1	o2	o3	o4	o5	-	
sparc/64	o0	o1	o2	o3	o4	o5	-	
tile	R00	R01	R02	R03	R04	R05	-	
x86-64	rdi	rsi	rdx	r10	r8	r9	-	
x32	rdi	rsi	rdx	r10	r8	r9	-	
xtensa	a6	a3	a4	a5	a8	a9	-	

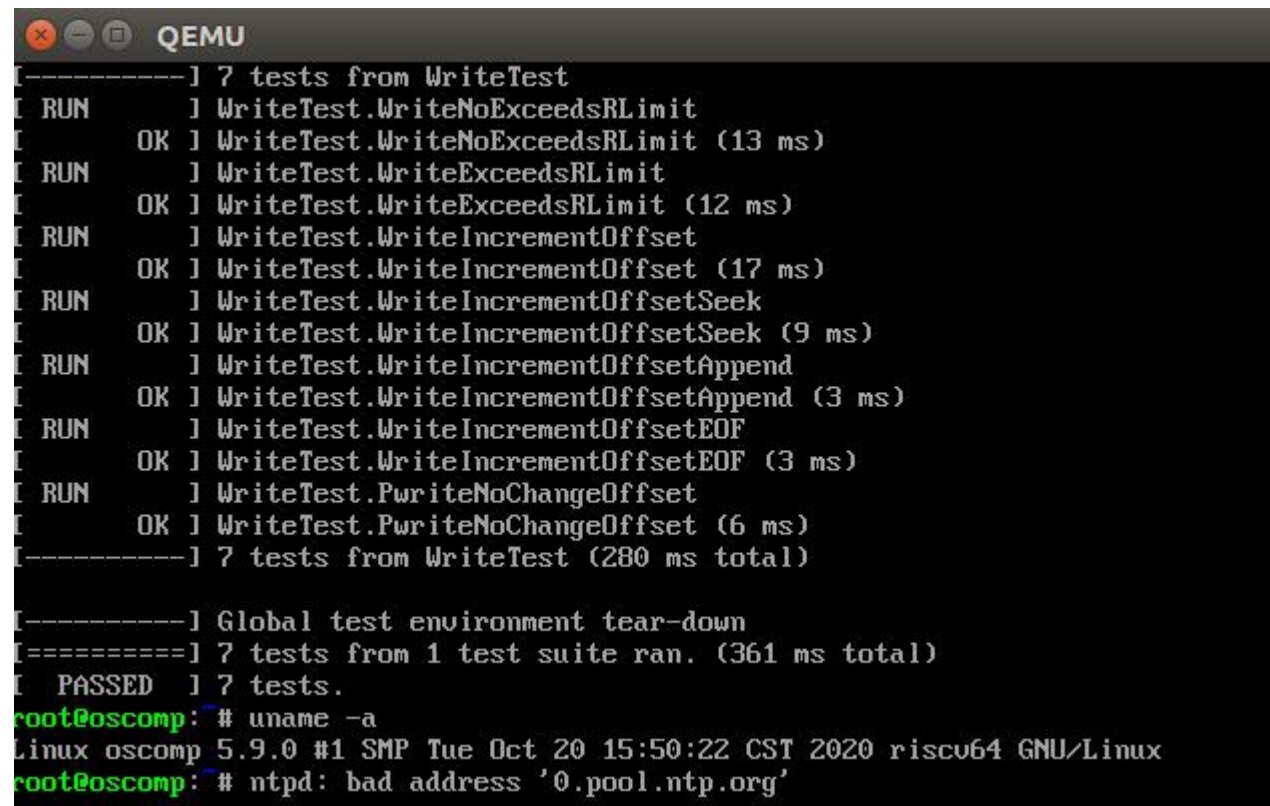
OS内核实现赛的测试用例

- 系统调用是用户态到内核态的入口点。
- 程序利用系统调用进行一系列操作，例如创建进程、处理网络、读写文件等。如当程序调用到 **open**、**read**、**write** 等函数时，就是在通过底层库来进行系统调用。
- 测试用例包括：
 - 内存管理
 - 进程管理
 - 信号处理
 - 进程间通信
 - 网络通信
 - 文件管理
 -

access	fanotify	getresuid	mkdir	pwrite	setgid	symlink
acct	fchdir	getrlimit	mkdirat	pwritev	setgroups	symlinkat
add_key	fchmod	get_robust_list	mknod	pwritev2	sethostname	sync
adjtimex	fchmodat	getrusage	mknodat	quotactl	setitimer	sync_file_range
alarm	fchown	getsid	mlock	read	set_mempolicy	syncfs
bind	fchownat	getsockname	mlock2	readahead	setns	syscall
bpf	fcntl	getsockopt	mlockall	readdir	setpgid	sysconf
brk	fdatasync	gettid	mmap	readlink	setpgrp	sysctl
cacheflush	fgetxattr	gettimeofday	modify_ldt	readlinkat	setpriority	sysfs
capget	flistxattr	getuid	mount	readv	setregid	sysinfo
capset	flock	getxattr	move_mount	realpath	setresgid	syslog
chdir	fmtmsg	inotify	move_pages	reboot	setresuid	tee
chmod	fork	inotify_init	mprotect	recv	setreuid	tgkill
chown	fpathconf	io_cancel	mq_notify	recvfrom	setrlimit	time
chroot	fremovexattr	ioctl	mq_open	recvmsg	set_robust_list	timer_create
clock_adjtime	fsconfig	io_destroy	mq_timedreceive	remap_file_pages	setsid	timer_delete
clock_getres	fsetxattr	io_getevents	mq_timedsend	removexattr	setsockopt	timerfd
clock_gettime	fsmount	ioperm	mq_unlink	rename	set_thread_area	timer_getoverrun
clock_nanosleep	fsopen	io_pgetevents	mremap	renameat	set_tid_address	timer_gettime
clock_settime	fspick	iopl	msync	renameat2	settimeofday	timer_settime
clone	fstat	ioprio	munlock	request_key	setuid	times
clone3	fstatat	io_setup	munlockall	rmdir	setxattr	tkill
close	fstatfs	io_submit	munmap	rt_sigaction	sgetmask	truncate
cma	fsync	io_uring	nanosleep	rt_sigprocmask	sigaction	ulimit
confstr	ftruncate	ipc	newuname	rt_sigqueueinfo	sigaltstack	umask
connect	futex	kcmp	nftw	rt_sigsuspend	sighold	umount
copy_file_range	futimesat	keyctl	nice	rt_sigtimedwait	signal	umount2
creat	getcontext	kill	open	rt_tgsigqueueinfo	signalfd	uname
delete_module	getcpu	lchown	openat	sbrk	signalfd4	unlink
dup	getcwd	lgetxattr	openat2	sched_getaffinity	sigpending	unlinkat
dup2	getdents	link	open_tree	sched_getattr	sigprocmask	unshare
dup3	getdomainname	linkat	paging	sched_getparam	sigrelse	userfaultfd
epoll	getdtablesize	listen	pathconf	sched_get_priority_max	sigsuspend	ustat
epoll_create1	getegid	listxattr	pause	sched_get_priority_min	sigtimedwait	utils
epoll_ctl	geteuid	llistxattr	perf_event_open	sched_getscheduler	sigwait	utime
epoll_pwait	getgid	llseek	personality	sched_rr_get_interval	sigwaitinfo	utimensat
epoll_wait	getgroups	lremovexattr	pidfd_open	sched_setaffinity	socket	utimes
eventfd	gethostbyname_r	lseek	pidfd_send_signal	sched_setattr	socketcall	vfork
eventfd2	gethostid	lstat	pipe	sched_setparam	socketpair	vhangup
execl	gethostname	madvise	pipe2	sched_setscheduler	sockioctl	vmsplce
execle	getitimer	Makefile	pivot_root	sched_yield	splice	wait
execlp	get_mempolicy	mallopt	pkeys	select	ssetmask	wait4
execv	getpagesize	mbind	poll	send	stat	waitid
execve	getpeername	membarrier	ppoll	sendfile	statfs	waitpid
execveat	getpgid	memcmp	prctl	sendmsg	statvfs	write
execvp	getpgrp	memcpy	pread	sendmsg	statx	writev
exit	getpid	memfd_create	preadv	sendto	stime	
exit_group	getppid	memmap	preadv2	setdomainname	string	

OS内核实现赛的测试用例DEMO

- 测试用例程序运行在QEMU模拟器的RISC-V Linux 5.9环境中；
- 测试用例展示测试功能名称，测试运行时间，测试运行成功与否，最后统计测试运行数量和PASSED数量；



```
QEMU
-----] 7 tests from WriteTest
RUN      1 WriteTest.WriteNoExceedsRLimit
OK      1 WriteTest.WriteNoExceedsRLimit (13 ms)
RUN      1 WriteTest.WriteExceedsRLimit
OK      1 WriteTest.WriteExceedsRLimit (12 ms)
RUN      1 WriteTest.WriteIncrementOffset
OK      1 WriteTest.WriteIncrementOffset (17 ms)
RUN      1 WriteTest.WriteIncrementOffsetSeek
OK      1 WriteTest.WriteIncrementOffsetSeek (9 ms)
RUN      1 WriteTest.WriteIncrementOffsetAppend
OK      1 WriteTest.WriteIncrementOffsetAppend (3 ms)
RUN      1 WriteTest.WriteIncrementOffsetEOF
OK      1 WriteTest.WriteIncrementOffsetEOF (3 ms)
RUN      1 WriteTest.PwriteNoChangeOffset
OK      1 WriteTest.PwriteNoChangeOffset (6 ms)
-----] 7 tests from WriteTest (280 ms total)

-----] Global test environment tear-down
=====] 7 tests from 1 test suite ran. (361 ms total)
PASSED   1 7 tests.
root@oscomp: # uname -a
Linux oscomp 5.9.0 #1 SMP Tue Oct 20 15:50:22 CST 2020 riscv64 GNU/Linux
root@oscomp: # ntpd: bad address '0.pool.ntp.org'
```

OS内核实现赛的测试用例DEMO

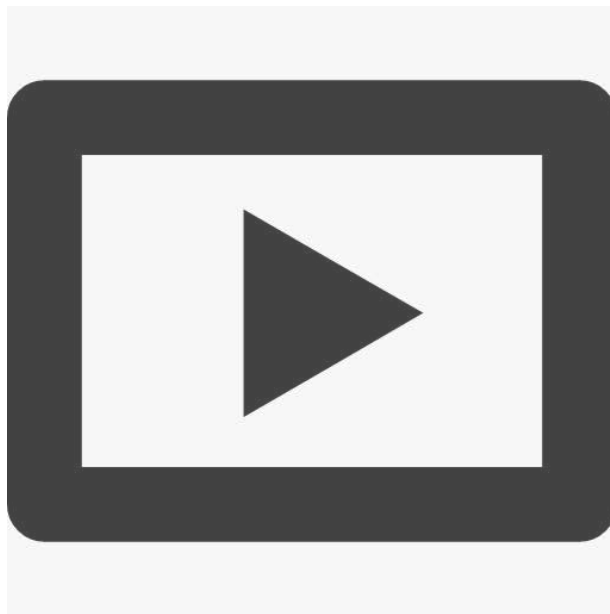
- 测试用例程序运行在K210开发板的RISC-V Linux 5.6环境中；
- 测试用例展示测试功能名称，统计测试运行是成功PASS或失败FAIL；

```
-----  
| Kendryte K210 Linux by OSCOMP |  
-----  
Mounting /proc  
Starting shell  
  
BusyBox v1.32.0.git (2020-12-21 13:43:54 CST) hush - the humble shell  
Enter 'help' for a list of built-in commands.  
  
/ # /root/testcases/ioctl01 -D /dev/tty9  
[ 56.781785] random: ioctl01: uninitialized urandom read (6 bytes)  
tst_test.c:1248: TINFO: Timeout per run is 0h 05m 00s  
ioctl01.c:71: TPASS: failed as expected: EBADF (9)  
ioctl01.c:60: TFAIL: call succeeded unexpectedly  
ioctl01.c:71: TPASS: failed as expected: ENOTTY (25)  
ioctl01.c:71: TPASS: failed as expected: ENOTTY (25)  
ioctl01.c:60: TFAIL: call succeeded unexpectedly  
/ #  
/ # uname -a  
Linux k210 5.6.0-rc1vowstar #28 SMP Fri Dec 25 12:52:45 CST 2020 ris
```

测试用例DEMO

DEMO环境:

- QEMU模拟器的RISC-V Linux
- K210开发板的RISC-V Linux



全国大学生OS比赛
期待您的参与
谢谢！