

# 2021 全国大学生计算机系统能力大赛操作系统赛内核实现赛道

参赛队伍：404 小队

队伍成员：吴亦泽，吕恒磊，王东宇

指导老师：蒋德钧

## 一、比赛准备和调研

查看决赛测试用例，调查用到的系统调用，参考 Linux 手册熟悉其功能和大致实现方法。我们查看了测试用例仓库，根据文件中列出的系统调用一一查找，对照在初赛实现的代码，查找尚未实现的系统调用。决赛用例中的系统调用均基于 Linux 系统调用给出，于是我们利用 Linux 手册，根据调用号和系统调用名一一查找比对，在着手实现前先做好标注，并根据系统调用的功能进行分类，便于后续工作的处理。

## 二、系统框架和模块设计

我们决赛阶段的系统内核沿用初赛的设计，框架已经基本构建好，决赛阶段的工作主要是继续实现系统调用，以及内核运行效率的优化。

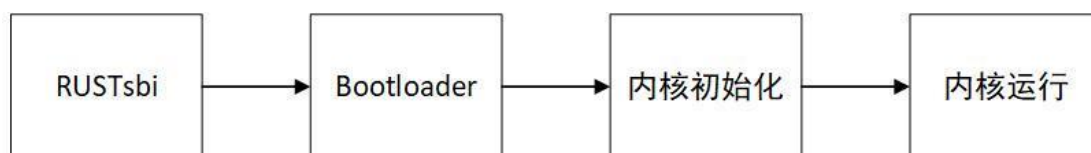


图 1

上图是我们操作系统内核的启动过程示意图。我们采用了官方提供的 RUSTsbi 作为开发板的启动部分，它会将我们实现的内核拷贝在内存的特定位置，并开始执行。之后是我们自行实现的操作系统加载模块，用于配置内核运行的环境、调整内核在内存中的位置、配置页表并开启虚拟地址访存模式等。内核部分开始运行时会先进行各个功能的初始化，包括进程管理模块初始化、中断处理初始化、系统调用初始化、计时器初始化、IO 初始化、内存交换初始化、SD 卡初始化、文件系统初始化等。完成所有的初始化过程后，我们的操作系统内核就能正式开始工作了。

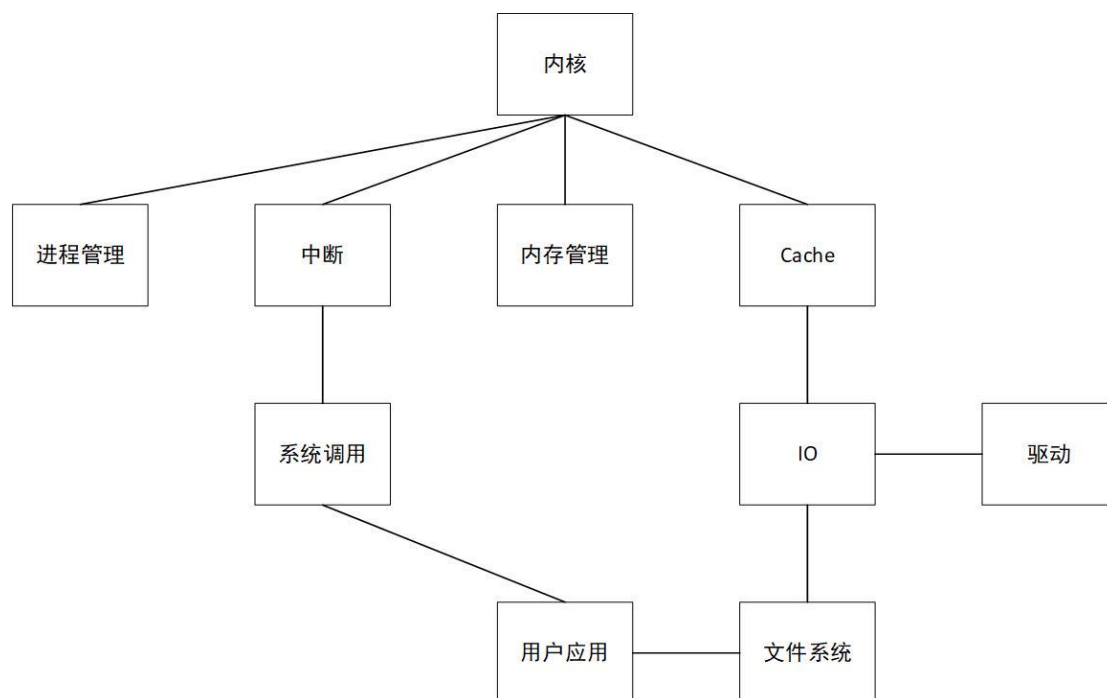


图 2

上图是我们的操作系统内核结构的示意图，显示了内核中的各个模块以及其间的关系。我们的内核主要有进程管理模块、中断处理模块、内存管理模块、IO 模块、驱动模块、文件系统模块等部分，各部分各司其职，分工协作。用户应用文件存放在文件系统中，内核借助驱动模块访问

磁盘，将用户应用加载到内存后开始执行。用户应用运行期间调用的系统调用会触发中断，陷入内核态进行处理，处理完成后返回用户态继续运行。

### 三、开发计划

决赛第一阶段正赶上学校期末考试，队员没有足够的时间进行操作系统内核开发，导致没能及时完成第一阶段的内容。期末考试结束后开始决赛第一和第二阶段的全部内容。

8月2日完成所有大三下学期的学习任务，开始准备比赛。经过一下午的头脑风暴，确认任务主要内容以及分工情况，开始开发，直至8月18日比赛结束。

### 四、比赛过程中的重要进展

8月3日，正确加载 debian 镜像中的 busybox 到内存；

8月4日，本地编译成功 busybox；

8月9日，实现按需分配内存；

8月11日，支持执行 sh 脚本；

### 五、遇到的主要问题和解决方法

首先是暑假期间小队三人没有聚在一起，而开发板只有两个，其中一个还时不时出现问题，导致只有一块板子能稳定进行开发。我们的解决方法是不具备硬件开发环境的同学在 qemu 上进行开发。

另外，决赛的第一阶段刚好与我们学校的期末考试时间冲突，队员无法匀出足够的时间进行内核开发。考试结束后，开始补足第一阶段的工作，但此时第一阶段的在线评测已经关闭，这给我们的测试带来了一定的困难。

我们在开发过程中使用 qemu 模拟硬件环境进行调试，但由于配置问题，我们的虚拟 SD 卡驱动一直没能做好，导致后期进行有关 IO 操作的内容的调试进行十分困难。虽然可以直接在开发板上打印日志调试，但调试的过程就变得极其繁琐。

我们在实现各个系统调用功能的时候也遇到过各种技术性问题，最终通过上网查阅资料以及向老师同学们请教讨论得以一一解决。

### 六、作品特征描述

优化：

1. 目录项的文件名查找使用逐字符比对。
2. 文件描述符的拷贝操作，不进行实质性的拷贝，而是利用 redirected 域，将新文件描述符指向原文件描述符，这样能节约空间以及数据拷贝时间；关闭描述符时，如果是新创建的则直接关闭，如果是初始的描述符，则将初始描述符的信息拷贝到重定向到它的某个文件描述符，并将所有重定向到初始描述符的描述符重定向至新拷贝的描述符。
3. do\_kill 系统调用的功能是一个进程向另一个进程发信号，可以是任何信号，不一定是 kill，这样可以实现更多功能。

### 七、分工和协作

吴亦泽完成了绝大部分系统调用的实现，王东宇负责 ppt 和文档编写，吕恒磊负责文件 io 的缓存设计

### 八、提交仓库目录和文件描述

<https://gitlab.eduxiji.net/HappyEric/oskernel2021-404>

```
.
├── arch      // boot 启动
│   └── riscv
│       ├── boot
│       ├── include
│       │   └── asm
│       └── kernel
```

```

|       └── sbi
|─── bootloader          // RUSTsbi
|─── drivers            // 驱动
|   └── sdcard
|       └── include
|─── img                // 文件系统镜像
|─── include            // 内核头文件
|   ├── os
|   ├── sched
|   ├── sys
|   ├── system
|   ├── time
|   └── utils
|─── init               // 内核初始化
|─── kernel             // 内核主体
|   ├── elf            // elf 文件处理
|   ├── fat32          // fat32 文件系统
|   ├── io             // IO
|   ├── irq            // 中断处理
|   ├── locking        // 锁的实现
|   ├── mm             // 内存管理
|   ├── sched          // 进程管理
|   ├── syscall        // 系统调用接口
|   ├── system         // 系统信息
|   └── time           // 时间相关
|─── libs              // 部分内核依赖库函数
|─── linker            // 链接器
|─── test              // 测试样例
|─── tiny_libc         // 用户态依赖库函数
|   └── include
|─── tools             // 依赖工具
|─── txt               // elf 文件反汇编结果，便于调试

```

## 九、比赛收获

这次比赛期间，我们遇到了各种各样的问题，包括时间管理、分工合作，以及代码完成和修正。另外，我们对操作系统内核的理解进一步加深，认识到了一个较为复杂的系统如何有序的运转。