

2024 ·  全国高校计算机系统能力培养 · OS 功能挑战赛道 · 华南区域赛

## Mit6.S081 4u<sup>1</sup>

–Proj0 面向操作系统课程的操作系统竞赛和实验<sup>2</sup>

–题目二：适合本校特点的实验指导流程

团队：mit 冲进决赛进中大<sup>‡</sup>

团队成员：蒋春鸿

---

<sup>1</sup>本文件采用 Typst 格式编写

<sup>2</sup>[https://compiler.educg.net/?token=7IolJqPnyovmyMsb7PInH7heiMaZokid1ofqAY9WM#/index?TYPE=OS\\_HNG](https://compiler.educg.net/?token=7IolJqPnyovmyMsb7PInH7heiMaZokid1ofqAY9WM#/index?TYPE=OS_HNG)

<sup>‡</sup>队伍 ID: T202410574995012; Project ID: 26035

# 大纲 (Outline)

1. 研究方案 (Plan)
2. 作品展示 (Exhibition)
3. 未来展望 (Prospect)
4. 问答 (Q&A)

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

# 研究方案 (Plan)

## 1. 研究方案 (Plan)

- 概述 (Overview)
- 背景 (Background)
- 应用定位 (Positioning)

## 2. 作品展示 (Exhibition)

## 3. 未来展望 (Prospect)

## 4. 问答 (Q&A)

### ► 研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

## 1.1 概述 (Overview)



关键词：(

- "操作系统",
- "教案",
- "实验指导文档",
- "实验要求",
- "参考实现",

)



**一段话概括团队工作：**整合了 MIT6.S081 项目中，官网的所有必做的实验 (lab) 的实验要求，给出具体实现步骤和过程当作教学教案，同时配套教案设计优化布局，提升学习体验。

这是一次为了使操作系统实验教学更能适应教学需求和学生理解，在全新艺术形态和表现手段上的尝试。

研究方案 (Plan)

► 概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

## 1.2 背景 (Background)

- 关于 Mit6.s 081<sup>1</sup>
  - Massachusetts Institute of Technology
  - 6: 电气工程与计算机科学 (EECS Electrical Engineering and Computer Science) 部门
  - S special, 081 无特殊含义唯一标识。前身是 6.828
  - 828 针对 x86, 项目命名为 JOS; 而本项目针对 RISC-V, 项目命名为 xv6 (基于 Unix ver6)

该项目在操作系统教育中占有非常高的地位

通过实践性的实验让学生深入理解各个核心概念

研究方案 (Plan)

概述 (Overview)

▸ 背景 (**Background**)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

---

<sup>1</sup><https://pdos.csail.mit.edu/6.828/2021/schedule.html>

6.S081: Operating System Engineering					2021
Schedule	Class	Labs	xv6	References	Piazza
Links to notes, videos etc. on future days are copies of materials from the 2020 version of 6.S081. We will update the notes as the course progresses. The lecture notes may help you remember the lecture content, but they are <i>not</i> a replacement for attending lectures.					
Monday	Tuesday	Wednesday	Thursday	Friday	
sep 6 Labor Day	sep 7 Reg Day	sep 8 <b>LEC 1 (ab):</b> Introduction and examples (handouts: xv6 book, 2020: notes, video) <b>Preparation:</b> Read chapter 1 (for your amusement: Unix) <b>Homework 1 due:</b> Question <b>Assignment:</b> Lab util: Unix utilities	sep 9	sep 10	
sep 13 <b>LEC 2 (TAs):</b> C and glibc <b>Preparation:</b> 2.9 (Bitwise operators) and 5.1 (Pointers and addresses) through 5.6 (Pointer arrays) and 6.4 (pointers to structures) by Kernighan and Ritchie (K&R)	sep 14	sep 15 <b>LEC 3 (ab):</b> OS organization and system calls (2020: notes, boards, video) <b>Preparation:</b> Read chapter 2 and xv6 code: kernel/proc.h, kernel/defs.h, kernel/entry.S, kernel/main.c, user/initcode.c, user/init.c, and skim kernel/proc.c and kernel/exec.c <b>Homework 2 due:</b> Question <b>Assignment:</b> Lab syscall: System calls	sep 16 <b>DUE:</b> Lab util	sep 17	
sep 20 <b>LEC 4 (ab):</b> Page tables (2020: notes, boards, video) <b>Preparation:</b> Read Chapter 3 and kernel/memlayout.h, kernel/vm.c, kernel/kalloc.c, kernel/iscv.h, and kernel/exec.c <b>Homework 3 due:</b> Question	sep 21	sep 22 <b>LEC 5 (TAs):</b> GDB, calling conventions and stack frames RISC-V (2020: notes, boards, video) <b>Preparation:</b> Read Calling Convention <b>Assignment:</b> Lab pgtbl: Page tables	sep 23 <b>DUE:</b> Lab syscall	sep 24	
sep 27 <b>LEC 6 (ab):</b> Isolation & system call entry/exit (2020: notes, video) <b>Preparation:</b> Read Chapter 4, except 4.6 and kernel/iscv.h, kernel/trampoline.S, and kernel/trap.c <b>Homework 4 due:</b> Question	sep 28	sep 29 <b>LEC 7 (ab):</b> Page faults (2020: notes, boards, video) <b>Preparation:</b> Read Section 4.6 <b>Homework 5 due:</b> Question <b>Assignment:</b> Lab traps: Traps	sep 30 <b>DUE:</b> Lab pgtbl	oct 1	
oct 4 <b>LEC 8 (ab):</b> Q&A labs <b>Homework 6 due:</b> Question	oct 5	oct 6 <b>LEC 9 (ab):</b> Interrupts (2020: notes, boards, video) <b>Preparation:</b> Read Chapter 5 and kernel/kernelsec.S, kernel/plc.c, kernel/console.c, kernel/uart.c, kernel/print.c <b>Homework 7 due:</b> Question <b>Assignment:</b> Lab cow: Copy-on-write fork	oct 7 <b>DUE:</b> Lab traps	oct 8 ADD DATE	
oct 11 Indigenous Peoples Day	oct 12	oct 13 <b>LEC 10 (ab):</b> Multiprocessors and locking (2020: notes, boards, video) <b>Preparation:</b> Read "Locking" with kernel/spinlock.h and kernel/	oct 14	oct 15	

研究方案 (Plan)

概述 (Overview)

► 背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

要下载实验材料

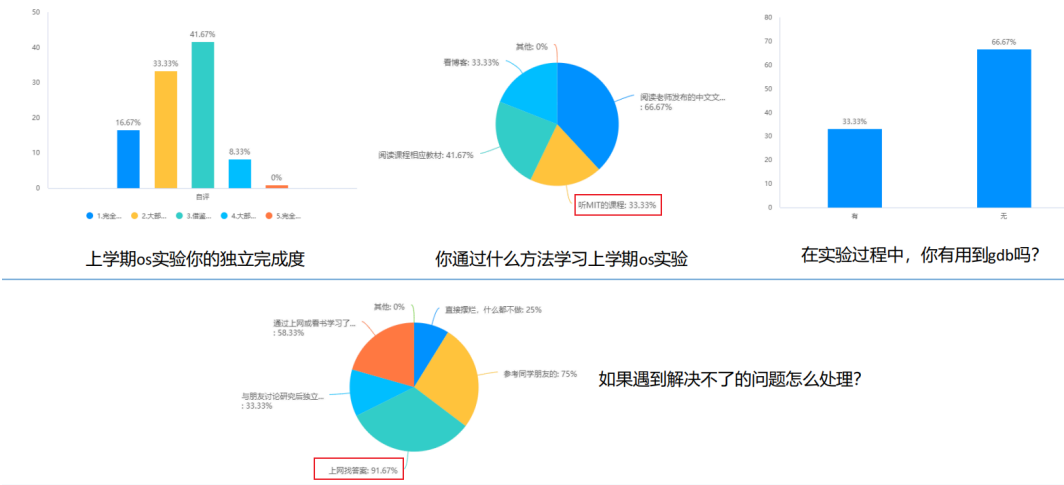
完成后要提交评测

界面（含题目）全是英文，对中文用户不友好

lab 之间相互堆叠，没有明确的界限

没有科学的实验步骤指导和成功率方案对比

调研<sup>1</sup>发现本校课程弊端显现



现有教案导致 OS 课程学习困难



研究方案 (Plan)

概述 (Overview)

► 背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

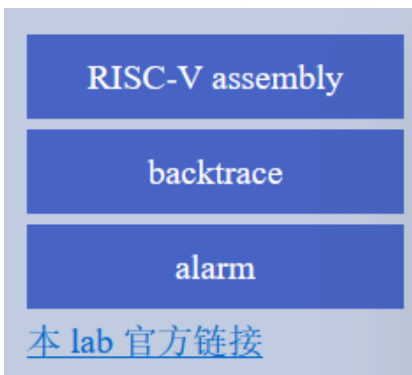
实验内各任务页面 (Task)

未来展望 (Prospect)

问答 (Q&A)

<sup>1</sup>选自本赛道本校 2021 年入围决赛队伍调研，队伍名：立\*\*\*\*\*队

- 改进实验指导文档
- 实验代码框架
- 实验用的测试用例
- 实验的参考实现



## 1.3 应用定位 (Positioning)

《一个集中式的在线 Mit6.S081 实验指导教学平台》

成品包括：一份教案 + 一个网站平台

在一般高校现有 OS 资源的基础上，进一步完成：

具有以下特点——

特点	体现
教学导向	实验设计注重理论与实践结合，强化理解和应用能力
可复现性	所有步骤都可以在标准环境中重现，确保学习体验
互动性强	鼓励学生通过解决实际问题来探索计算机科学深度
社区支持	项目链接至官网/外部资源，有丰富的教程供参考

### 要求

汇编理解，以 user/call.c 为例，通过 `make fs.img` 编译生成汇编程序 `user/call.asm`，阅读汇编程序，[官方参考资料](#)，熟悉 RISC-V 的一些汇编指令、寄存器等，回答问题：（下面用 A 来代替 Answer）

1. 存储参数的寄存器是哪个，`printf` 里传入值 13 放哪里？A: a0-a7, a2
2. 在哪里调用了函数 f 和 g？A: 没有这样的代码。g(x) 被内链到 f(x) 中，然后 f(x) 又被进一步内链到 main() 中
3. `printf` 的地址在哪？A: 0x0000000000000628, main 中使用 pc 相对寻址来计算得到这个地址。

研究方案 (Plan)

概述 (Overview)

背景 (Background)

► 应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)



# 作品展示 (Exhibition)

## 1. 研究方案 (Plan)

## 2. 作品展示 (Exhibition)

- 教案 (Teaching)
- 首页 (Index)
- 各实验首页 (Lab)
- 实验内各任务页面 (Task)

## 3. 未来展望 (Prospect)

## 4. 问答 (Q&A)

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

### ► 作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

# 2.1 教案 (Teaching)

## 《深入理解操作系统：MIT 6.S081课程精要》教案

《深入理解操作系统：MIT 6.S081课程精要》教案

写在前面

教学背景

项目背景

mit6.S081

x86, RISC-V

章节大纲

语法知识

.c .h 区别

第一章 环境部署

第二章 开发任务

lab1 util

sleep

要求

实现

pingpong

要求

实现

primes

要求

实现

find

要求

实现

xargs

要求

语法知识

.c .h 区别

c 文件

• 实现文件：这里包含了代码的实现部分，如函数的具体操作。

• 编译：.c 文件会被编译器编译成目标代码（.o 或 .obj 文件）。

• 执行代码：包含可执行的代码和逻辑。

h 文件

• 头文件：这里通常包含函数声明、类型定义、宏定义等。

• 不编译：.h 文件本身不会被编译，但会被包含（通过 #include 指令）在 .c 或其他 .h 文件中。

具体而言：

不会被编译

◦ 不生成目标代码：.h 文件本身通常不会直接被编译为目标代码（.o 或 .obj 文件）。

◦ 不直接参与链接：因为它们不能单独编译，所以也不会直接参与链接生成可执行文件或库文件的链接过程。

会被包含


◦ 文本替换：当你在一个 .c 或 .cpp 文件中用 #include "filename.h" 包含一个头文件时，编译器会将整个 .h 文件的内容文本性地“粘贴”到该位置。


◦ 声明可用：一旦一个头文件被包含，那么这个头文件中声明的所有函数、变量和类型定义就可以在包含它的 .c 或 .cpp 文件中使用。

• 接口和声明：.h 文件为编程者提供一个接口，使其知道可以使用哪些功能，但不需要知道这些功能是如何实现的。

第一章 环境部署

环境部署官方参考

 《Mit6.S081》操作系统实验教案.md

 《Mit6.S081》操作系统实验教案.pdf

- 使用 Typora 编写，.md 格式和.pdf 格式
- 与网页的教学资源同步配套
- 实验的参考实现：
  - ▶ **背景知识：**对相关概念的详细解释
  - ▶ **目标：**明确实验要达到的学习目标
  - ▶ **说明：**指导如何配置环境和执行任务的步骤
  - ▶ **代码：**使用 C 语言编写，让你直接动手实现操作系统级别的功能
  - ▶ **测试：**提供测试用例以验证你的实现是否正确

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

▶ 教案 (**Teaching**)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面 (Task)

未来展望 (Prospect)

问答 (Q&A)

## 2.2 首页 (Index)



- 点击上方导航栏的每个实验就可以进入每一个实验 (lab)
- 点击『[这里](#)』就可以下载配套的教案 (自编) 或官方的文档 (英文)

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

► 首页 (**Index**)

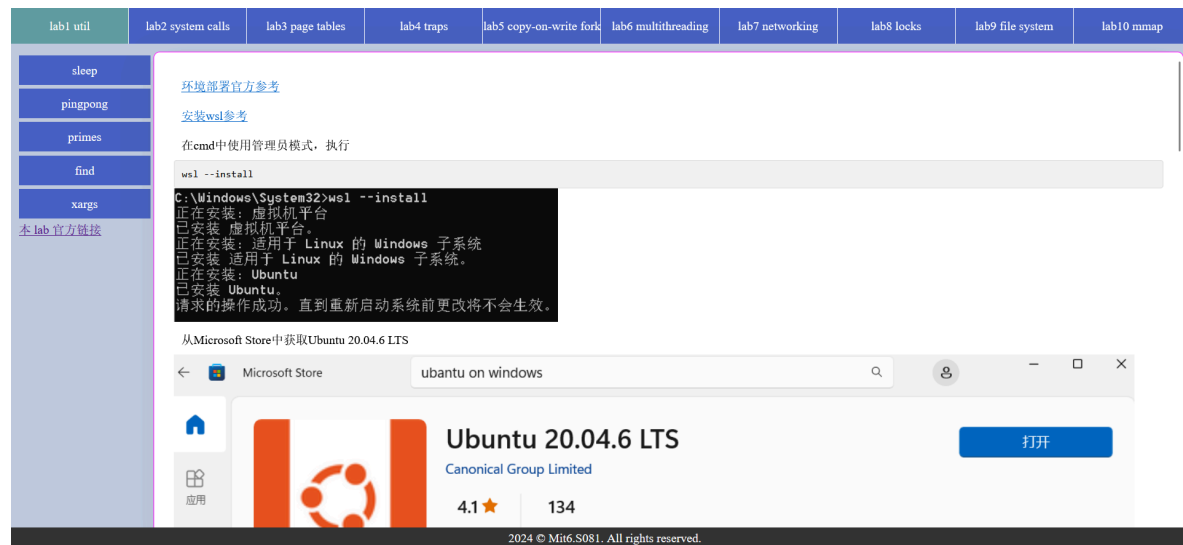
各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

## 2.3 各实验首页 (Lab)



研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

► 各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

- 一般是各个实验的前期准备，包括环境配置、需要了解的前置知识等

各实验包括：

## 1. util 实现用户态程序，熟悉系统调用使用

- sleep 进程休眠
- pingpong 两进程 ping 管道传输信息
- primes 进程创建子进程，父子通信
- find 遍历文件系统查找符合文件名要求的文件
- xargs 用管道连接多个命令，上一个输出当下一个输入

## 2. system calls 系统调用编写，理解用户态内核态切换等

- trace 执行一个用户程序，监听它用到了哪些系统调用并输出，以及中断保留现场的参数
- sysinfo 输出系统信息如空闲内存数、运行进程数。

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

► 各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

### 3. `page tables` 虚拟到物理内存映射的页表

- `speed up system calls` 在用户态和内核态间共享只读区域来加速系统调用 (`da ho`)
- `print a pagetable` 打印页表及其子页表
- `detecing which pages have been accessed` 查询自上次查询以来，有多少页面被访问过

### 4. `traps` 中断处理机制

- `RISC-V assembly` 阅读代码理解对应的汇编代码
- `backtrace` 输出调用栈的所有函数返回地址
- `alarm` 执行自定义周期函数

### 5. `copy-on-write fork COW` 机制，延迟复制页直到需要写

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

► 各实验首页 (**Lab**)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

## 6. multithreading 多线程

- **Uthread: switching between threads** 用户级多线程间的上下文切换
- **Using threads** 使用多线程优化哈希表, 使用桶级锁
- **Barrier** 实现检查点(线程屏障)

## 7. networking 实现网络收发包

## 8. locks 细化锁粒度优化并发性能

- **Memory allocator** 优化内存分配, 将锁粒度从整个分配器精细到各 CPU 级; 每个内存由单个 CPU 独占
- **Buffer cache** 磁盘文件高速缓存块从整个分配器锁细化到分哈希表每个桶一个锁; 涉及页面置换, 二次探测等并发编程技巧

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

► 各实验首页 (**Lab**)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)

## 9. file system

- **large files** 修改一个直接索引块为二级间接块，提升单个文件的最大容量
- **symbolic links** 给文件创建符号链接/软链接

10. **mmap** 将一段磁盘文件地址懒映射到内存地址，在需要时读盘取数据

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

► 各实验首页 (**Lab**)

实验内各任务页面  
(Task)

未来展望 (Prospect)

问答 (Q&A)



### 研究方案 (Plan)

## 概述 (Overview)

## 背景 (Background)

## 应用定位 (Positioning)

### 作品展示 (Exhibition)

教案 (Teaching)

[首页 \(Index\)](#)

各实验首页 (Lab)

- ▶ 实验内各任务页面  
(Task)

### 未来展望 (Prospect)

## 问答 (Q&A)

`fork()` 系统调用复制所有父进程用户空间内存给子进程。可能是浪费的，因为子不一定要用全部内存，但又不能完全共享，因为会写。

使用 COW(copy-on-write) 机制，延迟复制到需要写。在该机制下，创建节点页页时，PTE 里的用户内存直接指向父的，且这样的指针标记为不可写的。父或子需要写时，将产生页错误，被处理错误后，然后进行复制分配，调整 PTE，标记为可写的，返回现场。

使用 COW 时，释放内存较为麻烦，一个页可能被子多个指针指向，只有所有指针不再指向时才能释放内存。

需要通过 `contest` 和 `usertests`。在不实现 COW 时，会因为 MLE 无法通过测试。

示例:

```
$ connect
single: on
single: on
three: combine
ok
three: combine
ok
three: combine
ok
file: ok
ALL COW TESTS PASSED
$ userhosts
...
ALL TESTS PASSED
$
```

50. 2014 年 10 月 1 日

組織系圖 3.3 中，可知給予 A 本區辦公，可得到一本外來公文數目是型 6000 的公文。

張大勇、潘如雲、王

$$\text{Definition DTE } C(A, B) \text{ if } C \text{ can be written}$$

在 `mainloop.h` 里, 有常量 `PHYSTOP` 代表最大物理地址, 在 `page.h` 有 `PAGESZ` 代表页大小, 相除得总页数 `557056`, 需要的 `2MB` 的大小开 `int` 数组, 所以可以开一个全局数组记录每个页编号是否占用, 考虑内存浪费, 再开一个块, 在 `kernel.c` 里:

```
int refcount(FIRSTOP/NGST
```

© 2006 The Authors  
Journal compilation © 2006 Blackwell Publishing Ltd

```
int po = po+PHYSIZE; //physical number
acquire(&mem_lock);
if((po>PHYSIZE || refcount[po]<1){
    panic("incret");
}
refcount[po] += 1;
release(&mem_lock);
```

## 效果

```
# sleep 10
$ QEMU: Terminated
● root@LAPTOP-5SDGUMAS:/xv6-labs-2021# ./grade-lab-util sleep
make: 'kernel/kernel' is up to date.
== Test sleep, no arguments == sleep, no arguments: OK (1.4s)
== Test sleep, returns == sleep, returns: OK (0.9s)
== Test sleep, makes syscall == sleep, makes syscall: OK (0.9s)
○ root@LAPTOP-5SDGUMAS:/xv6-labs-2021#
```

## 要求

官方要求: Implement the UNIX program `sleep` for xv6; your `sleep` should pause for a user-specified number of ticks. A tick is a notion of time defined by the xv6 kernel, namely the time between two interrupts from the timer chip. Your solution should be in the file `user/sleep.c`.

**1.进程和内存：** 每个进程都拥有自己的用户空间内存以及内核空间状态，当进程不再执行时，xv6会存储和这些进程有关的CPU寄存器到下一次运行这些进程。kernel中一个进程有唯一的PID。常用的syscall：

- `fork`: 原型是 `int fork()`。作用是让一个进程生成另一个和这个进程的内存内容相同的子进程。在父进程中, `fork`的返回子进程的PID, 在子进程中, 返回值是0。
  - `exit`: 原型 `int exit(int status)`。作用是让调用它的进程停止执行并且将内存等占用的资源全部释放。`status`是状态参数, 0代表正常退出, 1代表非正常退出。
  - `wait`: 原型 `int wait(int *status)`。等待子进程退出, 返回子进程PID, 子进程的退出状态存储到`*status`地址中。如果没有调用子进程, `wait`返回-1。
  - `exec`: 原型 `int exec(char *file, char *argv[])`。作用是加载一个文件, 获取执行它的参数, 执行。执行错误返回-1, 执行成功则不会返回, 而开始从文件入口位置开始执行命令, 文件格式必须是ELF格式。
- ## 2.IO和文件描述符
- `file` 描述符: 文件描述符, 一个被内核管理的、可以被进程读、写的对象的一个整数, 通过打开文件、目录、设备等方式获得。一个文件被打开的越早, 文件描述符越小, 每个进程都有自己独立的文件描述符列表, 0是标准输入, 1是标准输出, 2是标准错误。`shell`保证总是3个文件描述符是可用的, 在给的源码中的`sh.c`中有这样一段代码

```
int fd;

// Ensure that three file descriptors are open.
while((fd = open("console", O_RDWR)) >= 0){
    if(fd >= 3){
        close(fd);
        break;
    }
}
```

2024 © MIT6.S081. All rights reserved.

- 包含要求、实现和效果三个部分
- 要求是将官网的题目要求翻译成中文
- 实现是具体的修改和实现步骤（含代码）
- 效果是最终每一个任务的正确性验证截图，方便比对

# 未来展望 (Prospect)

1. 研究方案 (Plan)
2. 作品展示 (Exhibition)
- 3. 未来展望 (Prospect)**
4. 问答 (Q&A)

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

► **未来展望 (Prospect)**

问答 (Q&A)

# 总结 (Conclusion)

- 目前测试反馈使用效果良好
- 将网站部署到服务器，用于学校日常课程教学，且迭代运维
- 增加实践案例与项目，丰富教案与课程资源

了解更多	文档	代码/视频
实验平台	<a href="#">作品设计报告.pdf</a> <a href="#">实验平台简介.pdf</a>	<a href="#">演示视频</a>
实验代码上传者	<a href="#">代码上传者.pdf</a>	<a href="#">代码上传者代码-uploader</a>
实验平台后端_评测主机	<a href="#">实验平台后端_评测主机.pdf</a>	<a href="#">评测主机代码-host</a>
实验平台后端_评测从机	<a href="#">实验平台后端-评测从机.pdf</a>	<a href="#">评测从机代码-worker</a>
实验平台前端	<a href="#">实验平台前端.pdf</a>	<a href="#">前端代码-frontend</a>
其他	<a href="#">实验指导书部署指南</a>	

- 关于评测机功能的改进，增强互动参与感
- 个性化学习路径（注册登录/个性推荐）

[研究方案 \(Plan\)](#)

[概述 \(Overview\)](#)

[背景 \(Background\)](#)

[应用定位 \(Positioning\)](#)

[作品展示 \(Exhibition\)](#)

[教案 \(Teaching\)](#)

[首页 \(Index\)](#)

[各实验首页 \(Lab\)](#)

[实验内各任务页面 \(Task\)](#)

[未来展望 \(Prospect\)](#)

[问答 \(Q&A\)](#)

## 问答 (Q&A)

1. 研究方案 (Plan)

2. 作品展示 (Exhibition)

3. 未来展望 (Prospect)

4. 问答 (Q&A)

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

► 问答 (Q&A)

# 问答 (Q&A)

1. 感谢赛方<sup>1</sup>的帮助
2. 感谢 Typst<sup>2</sup>、NewCM<sup>3</sup>、Noto<sup>4</sup>
  - Slides 的字体采用 NewCM 和 Noto
3. 欢迎老师提问

研究方案 (Plan)

概述 (Overview)

背景 (Background)

应用定位 (Positioning)

作品展示 (Exhibition)

教案 (Teaching)

首页 (Index)

各实验首页 (Lab)

实验内各任务页面  
(Task)

未来展望 (Prospect)

► 问答 (Q&A)

---

<sup>1</sup><https://os.educg.net/#/>

<sup>2</sup><https://github.com/typst/typst>

<sup>3</sup><https://git.gnu.org.ua/newcm.git>

<sup>4</sup><https://fonts.google.com/noto>