从零构建操作系统-学生指导手册

总说明

开发环境要求

- **推荐开发平台**：Ubuntu 22.04 LTS（或 WSL2）
- **基础工具**：git / make / Python3 / qemu-system-riscv64 / riscv64-unknown-elf-gcc
- **学习方法**：理解原理 → 参考xv6 → 独立实现 → 测试调试
- **进度要求**：每完成一个阶段运行测试并提交代码

核心学习资料

- **xv6-riscv源码**：https://github.com/mit-pdos/xv6-riscv
- **RISC-V规范**：https://riscv.org/technical/specifications/
  - 重点：Volume II: Privileged Specification（特权级规范）https://drive.google.com/file/d/17GeetSnT5wW3xNuAHI95-SI1gPGd5sJ_/view
  - 在线版本：https://riscv.github.io/riscv-isa-manual/snapshot/privileged/#_preface
- **xv6手册**：https://pdos.csail.mit.edu/6.828/2025/xv6/book-riscv-rev5.pdf

统一提交要求

每个实验需提交以下内容：

```
1. 源代码仓库（通过git管理）
   ├── 完整实现代码（带注释）
   ├── Makefile
   ├── README.md（编译运行说明）
   └── 规范的目录结构

2. 综合实验报告（report.md）
   ├── 系统设计部分
   │   ├── 架构设计说明
   │   ├── 关键数据结构
   │   ├── 与xv6对比分析
   │   └── 设计决策理由
   ├── 实验过程部分
   │   ├── 实现步骤记录
   │   ├── 问题与解决方案
   │   └── 源码理解总结
```

```
    └── 测试验证部分
        ├── 功能测试结果
        ├── 性能数据
        ├── 异常测试
        └── 运行截图/录屏
```

实验0：开发环境搭建

环境搭建步骤

1. 安装基础依赖

```
# Ubuntu/Debian系统
sudo apt-get update
sudo apt-get install -y build-essential git python3 qemu-system-misc expect gdb-multiarch

# 验证QEMU版本（建议5.0+）
qemu-system-riscv64 --version
```

2. 安装RISC-V工具链

```
# 方案A：使用预编译包（推荐）
wget https://github.com/riscv-collab/riscv-gnu-toolchain/releases/download/2023.07.07/riscv64-elf-ubuntu-20.04-gcc-nightly-2023.07.07-nightly.tar.gz
sudo tar -xzf riscv64-elf-ubuntu-20.04-gcc-nightly-2023.07.07-nightly.tar.gz -C /opt/
echo 'export PATH="/opt/riscv/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc

# 方案B：包管理器安装（可能版本较旧）
sudo apt-get install gcc-riscv64-unknown-elf

# 验证安装
riscv64-unknown-elf-gcc --version
```

3. 获取参考资料

```
# 克隆xv6源码作为参考
git clone https://github.com/mit-pdos/xv6-riscv.git
cd xv6-riscv && make qemu  # 验证能否正常运行
```

4. 创建项目结构

```
mkdir riscv-os && cd riscv-os
git init
mkdir -p kernel/{boot,mm,trap,proc,fs,net} include scripts
```

5. 验证环境

创建测试文件验证交叉编译：

```
echo 'int main(){ return 0; }' > test.c
riscv64-unknown-elf-gcc -c test.c -o test.o
file test.o  # 应显示RISC-V 64-bit
```

实验1：RISC-V引导与裸机启动

实验目标

通过参考xv6的启动机制，理解并实现最小操作系统的引导过程，最终在QEMU中输出"Hello OS"。

核心学习资料

xv6关键文件分析

- `kernel/entry.S` - 启动汇编代码
  - 重点理解：栈设置、BSS清零、跳转到C代码
  - 思考：为什么需要设置栈？栈应该多大？
- `kernel/kernel.ld` - 链接脚本
  - 重点理解：入口点设置、段的组织、符号定义
  - 思考：各个段的作用和排列顺序
- `kernel/uart.c` - 串口驱动
  - 重点理解：UART寄存器操作、字符输出实现
  - 思考：如何简化为最小实现？

RISC-V启动机制

- **特权级规范第3.1节**：机器模式启动状态
- **QEMU virt平台**：内存布局和设备地址

```
# 查看QEMU设备树
qemu-system-riscv64 -machine virt,dumpdtb=virt.dtb -nographic
dtc -I dtb -O dts virt.dtb | grep -A5 -B5 "uart\|memory"
```

任务列表