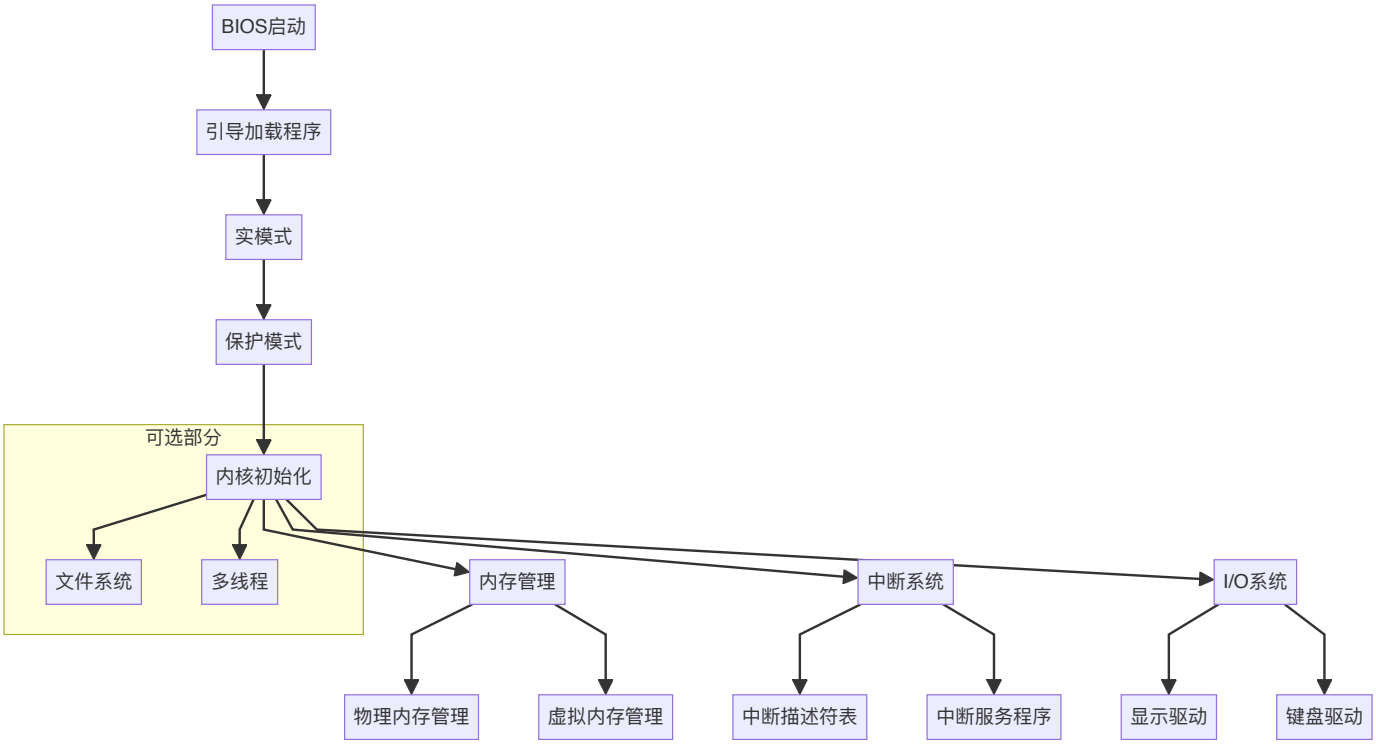
设计文档

陈佩乐

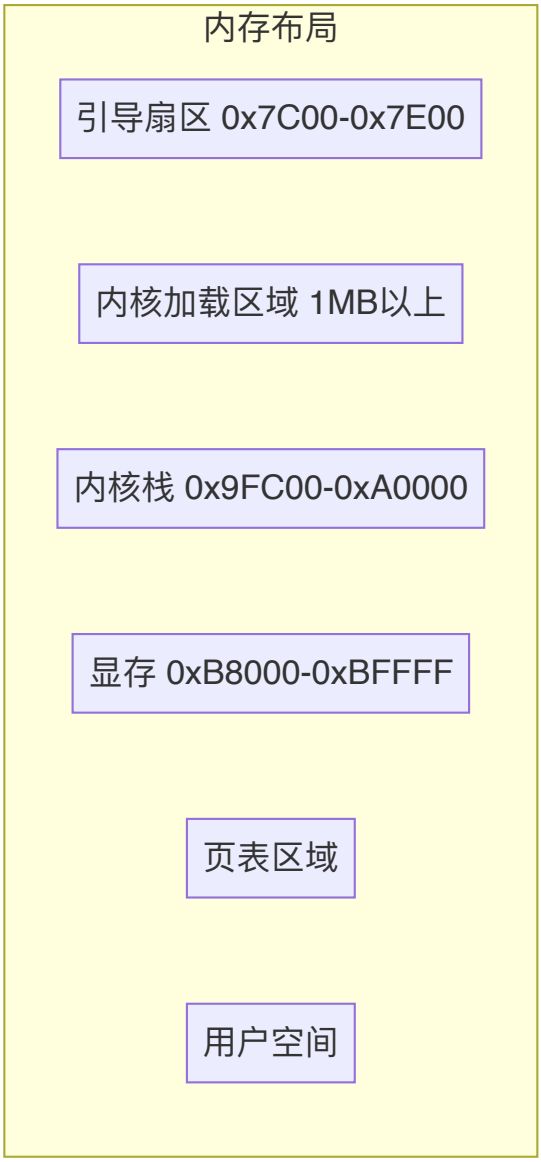
简化版OS Demo设计文档

1 1. 系统架构概述

1.1 1.1 整体架构



1.2 1.2 内存布局



2 2. 详细设计

2.1 2.1 引导阶段

2.1.1 2.1.1 引导扇区设计

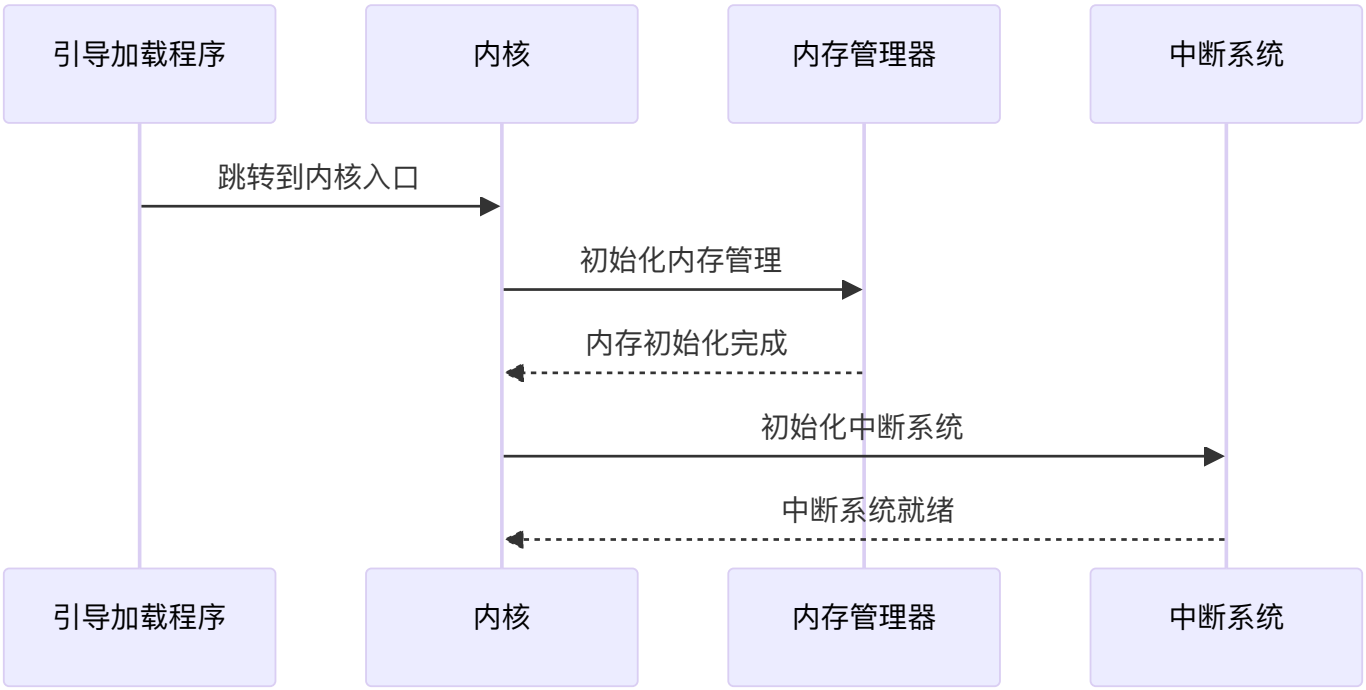
- 大小：512字节
- 功能：
  1. 初始化段寄存器
  2. 设置栈指针
  3. 加载内核到内存
  4. 准备切换到保护模式

2.1.2 2.1.2 GDT设计

```
struct gdt_entry {
    uint16_t limit_low;     // 段界限 15:0
    uint16_t base_low;      // 段基址 15:0
    uint8_t  base_middle;   // 段基址 23:16
    uint8_t  access;        // 访问标志
    uint8_t  granularity;   // 颗粒度
    uint8_t  base_high;     // 段基址 31:24
} __attribute__((packed));
```

2.2 2.2 内核设计

2.2.1 2.2.1 内核初始化流程



2.2.2 2.2.2 中断系统设计

- IDT表项结构：

```
struct idt_entry {
    uint16_t offset_low;    // 处理程序地址低16位
    uint16_t selector;      // 代码段选择子
    uint8_t  zero;          // 保留
    uint8_t  type_attr;     // 类型和属性
    uint16_t offset_high;   // 处理程序地址高16位
} __attribute__((packed));
```

- 中断处理流程：
  1. 保存现场
  2. 调用中断处理程序
  3. 恢复现场
  4. 中断返回

2.3 2.3 驱动层设计

2.3.1 2.3.1 显示驱动

- 显存地址：0xB8000
- 支持功能：
  1. 字符显示
  2. 光标控制
  3. 滚屏
  4. 颜色控制

2.3.2 2.3.2 键盘驱动

- 中断号：IRQ1
- 功能实现：
  1. 扫描码转换
  2. 按键缓冲区
  3. 特殊键处理

2.4 2.4 Shell设计

2.4.1 2.4.1 命令处理流程

2.4.2 2.4.2 基本命令集

1. help - 显示帮助信息
2. clear - 清屏
3. info - 显示系统信息
4. reboot - 重启系统
5. shutdown - 关闭系统

2.5 2.5 可选功能设计

2.5.1 2.5.1 文件系统

- 简单文件系统结构：

```
struct file_entry {
    char name[32];
    uint32_t size;
    uint32_t flags;
    uint32_t start_block;
};
```

2.5.2 2.5.2 多线程支持

- 线程控制块设计：

```
struct thread_control_block {
    uint32_t esp;       // 栈指针
    uint32_t ebp;       // 基址指针
    uint32_t eip;       // 指令指针
    uint32_t status;    // 线程状态
    uint32_t priority; // 优先级
    uint32_t ticks;     // 时间片
};
```

3 3. 接口设计

3.1 3.1 内核API

```
// 内存管理
void* kmalloc(size_t size);
void kfree(void* ptr);

// 中断相关
void register_interrupt_handler(int n, interrupt_handler_t handler);
void enable_interrupt(void);
void disable_interrupt(void);

// 输入输出
void printf(const char* format, ...);
char getchar(void);
void putchar(char c);
```

3.2 3.2 驱动接口

```
// 显示驱动
void init_video(void);
void clear_screen(void);
void set_cursor(int x, int y);

// 键盘驱动
void init_keyboard(void);
int read_scancode(void);
char translate_scancode(int scancode);
```

4 4. 开发规范

4.1 4.1 代码组织

```
/
├── boot/
│   ├── boot.asm
│   └── loader.asm
├── kernel/
│   ├── init/
│   ├── mm/
│   ├── interrupt/
│   └── drivers/
├── include/
```

```
│   ├── kernel/
│   └── drivers/
└── userland/
    └── shell/
```

4.2 4.2 编码规范

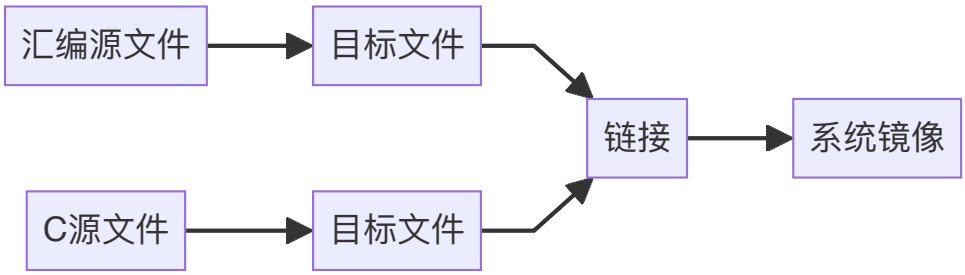
- 使用K&R缩进风格
- 函数名使用小写字母加下划线
- 常量和宏使用大写字母
- 每个文件开头添加版权声明和功能描述
- 关键函数添加详细注释

5 5. 构建系统

5.1 5.1 工具链要求

- NASM >= 2.13
- GCC >= 7.3.0
- GNU Make >= 4.2
- QEMU >= 2.11.1

5.2 5.2 构建流程



6 6. 测试计划

6.1 6.1 单元测试

- 内存管理测试
- 中断处理测试
- 设备驱动测试
- Shell命令测试

6.2 6.2 集成测试

- 启动流程测试
- 系统功能测试
- 压力测试
- 稳定性测试

7 7. 部署方案

7.1 7.1 开发环境部署

1. 安装必要的开发工具
2. 配置编译环境
3. 设置QEMU测试环境

7.2 7.2 实机部署

1. 创建启动盘
2. 复制系统镜像
3. 设置BIOS启动顺序