操作系统项目工程文档

基于自制内核

Your Name

2025 年 11 月 7 日

目录

前言

这是一套基于 **ctexbook** 与自定义样式的工程/教材级文档骨架，目标是：

- 美观、易读，适合“书籍式”通读；- 代码高亮丰富（优先 `minted`，可自动降级到 `listings`）；- 结构清晰，便于长期维护与扩展；- 与仓库保持紧密联系，支持直接引用源码文件。

> **快速开始**
>
> 在 `doc/` 目录执行：`latexmk`；PDF 输出位于 `_build/main.pdf`。若系统缺少 `minted`，将自动使用 `listings`。

第一章　引导与磁盘布局

本章概览磁盘镜像、MBR 与 Loader 的位置，以及从实模式到保护模式的关键步骤。为示例起见，下面展示一个极简的 MBR 代码片段（仅示意）。

1.1　极简 MBR 片段

```
; demo only — not a real boot sector
BITS 16
org 0x7C00
cli
xor ax, ax
mov ds, ax
mov es, ax
sti
hlt
times 510-($-$$) db 0
dw 0xAA55
```

1.2　引用仓库源码

可以直接从仓库中引入源码并自动高亮，例如：

```


export module os;

import kernel;
import test.filesystem;

auto func() -> void
{
    auto a = std::vector(50000,int{});
    for(auto i = 0; i =/= a.size(); ++i) {
        a[i] = i;
    }
```

```
    auto fd = open("/test_f",+open_flags::create |
    ↪  +open_flags::write);
    write(fd,a.data(),a.size());
    close(fd);
    fd = open("/test_f",+open_flags::read);
    auto b = std::vector(50000,int{});
    read(fd,b.data(),b.size());
    for(auto i = 0; i =/= 20; ++i) {
        console::print("{} ",b[i]);
    }
    console::println();
}

auto main() -> void
{
    func();
}
```

如需引用汇编：`\codefile{nasm}{../boot/mbr.asm}`（路径请按你的仓库实际调整）。

第二章　内存管理

本章将逐步介绍内核内存布局、物理页分配器、分页结构与内核堆的设计权衡与实现要点。以下为一个 C++ 伪代码片段示例：

示例：物理页分配接口

```cpp
struct page {
    unsigned long pfn; // page frame number
    bool used;
};

auto alloc_page() -> page* {
    // TODO: implement buddy or bitmap allocator
    return nullptr;
}

auto free_page(page* p) -> void {
    // TODO: recycle
}
```

第三章　样式与组件总览（Sampler）

本章汇集了常用的版式元素，便于快速检查“教材风”样式在你的环境下的呈现效果：标题、段落、列表、代码高亮、盒子、表格、长表格与 TikZ 简图等。

3.1　标题层级与段落

这是一个普通段落。中文采用宋体（Noto Serif CJK SC），英文采用 Libertinus Serif。默认首行缩进两字符，段间距为 0，行距 1.25。Inline code 示例：`mov ds, ax`、`0x7C00`。

3.1.1　小节标题

小节风格紧凑，不占过多垂直空间。再来一个段落，用于观察换行与对齐效果。

小小节标题

说明性文本，继续观察标题与正文之间的间距与对比度是否符合“教材感”。

3.2　列表

- 无序列表项 A
- 无序列表项 B

1. 有序列表项 1
2. 有序列表项 2

3.3　提示/警告盒子

注意

使用 `\notebox` 呈现普通提示；白底、浅灰边。可放入简短定义或引导性说明。

警告

用于重要风险提示，例如“写盘会覆盖镜像”。

提示

用于辅助信息或技巧，比如“Bochs 日志可用 `\codefile` 引入片段用于记录实验证据”。

3.4　代码高亮（minted）

3.4.1　C++ 片段

示例：C++

```cpp
#include <cstdint>
auto sum(int a, int b) -> int {
    return a + b;
}
```

3.4.2　NASM 片段

示例：NASM

```asm
BITS 16
org 0x7C00
cli
xor ax, ax
mov ds, ax
sti
```

3.4.3　引用仓库文件（前 40 行）

```asm


[bits 32]
%define ERROR_CODE nop  ; 主要用于统一对齐栈，不压入 ERROR_CODE 的中断程序就压入 0
%define ZERO push 0

extern puts
extern idt_table ; C 中注册的中断处理程序数组
extern syscall_table

global intr_entry_table
global intr_exit
global syscall_handler

section .data

```

```
intr_str db "interrupt occur!", 0xA, 0
intr_entry_table:

%macro VECTOR 2
section .text
intr%1entry:
    %2
    push ds
    push es
    push fs
    push gs
    pushad


    mov al, 0x20
    out 0xa0, al     ; 向从片发送 EOI
    out 0x20, al     ; 向主片发送 EOI

    push %1          ; 压入中断向量号
    call [idt_table + %1 * 4]
    jmp intr_exit

section .data
    dd intr%1entry   ; 中断入口程序地址
```

3.5　表格与长表格

3.5.1　普通表格

符号	说明
GDT	全局描述符表
IDT	中断描述符表
TSS	任务状态段

表 3.1: 核心数据结构缩略语

3.5.2　长表格（示例）

表 3.2: 中断向量片段（示意）

向量	类型	说明
0x00	异常	除零错误
0x01	异常	调试
0x20	外设	PIT 定时器
0x21	外设	键盘
0x80	系统调用	用户态陷入

3.6　TikZ 简图：内存布局（示意）