Simple-XX 社区项目：为 SimpleKernel 编写项目文档 申请书

项目情况

SimpleKernel 项目将内核开发分为一下下七个部分：

1. boot

   系统启动

2. printf

   调试输出

3. pmm

   物理内存管理

4. vmm

   虚拟内存管理

5. heap

   堆管理

6. lib

   c++ 库的移植

7. intr

   中断管理

项目的任务是针对以上 branch 介绍原理与代码实现，粒度为每个文件，同时整理相关资料，最后还需要有英文版。

实现方案

我将项目分为以下几个部分

1. 启动

   这部分讲引导代码的通用思路，再针对引导程序（如 grub、opensbi）进行特别讲解。

2. 调试输出

   说明 TUI、sbi_console 的原理与使用方法。

3. 内存初始化

   我将物理内存、虚拟内存与堆三个部分放在一起。

   物理内存部分，主要说明如何获取物理内存信息（x86 下为通过 multiboot2 规范获取相关信息）。

   虚拟内存部分，主要说明虚拟内存在不同架构下的不同实现细节，比如对于 riscv，就要着重讲解 `pte_t *walk(pt_t pgd, uint64_t va, bool alloc)` 函数的作用。

   堆部分，需要说明 malloc/free 与 new/delete 相关的细节（heap.cpp 与 new.cpp）。

4. 库的移植

   主要说明 src/libcxx 下每个文件的用法，说明 cxxabi.cpp 的细节，说明 `void cpp_init(void)` 是如何工作的。

5. 文件系统初始化

   SimpleKernel 目前实现了一个简单的 RAMFS（MRNIU/SimpleKernel: vfs），这里主要说明这个简单文件系统的细节。

- 对于所有内容，我会首先描述这个模块的抽象思路，并尽可能在代码中添加 API 的注释。
- 需要的参考资料

  https://wiki.osdev.org/Main_Page

  Intel 手册

  multiboot2 规范

  grub2 手册

  riscv 手册

  C++ std 相关知识

时间安排

时间	安排
七月 第一周～第四周	熟悉项目代码，为代码添加注释
八月 第五周～第八周	从整体视角对模块进行描述
九月 第九周～第十二周	总结前两周的工作，整理后将文档以 Markdown 的形式存放在相应目录下，如果时间足够，产出相应的英文版

示例

以 Simple-XX/SimpleKernel: boot 下的 i386 架构的引导代码为例。

```
# 版权信息
# This file is a part of Simple-XX/SimpleKernel (https://github.com/Simple-XX/SimpleKernel).
#
# boot.S for Simple-XX/SimpleKernel.

# 添加定义，这个文件中定义了 Multiboot2 使用的宏
#include "multiboot2.h"

# 说明这段代码是 32 位的
```

```
.code32

# multiboot2 文件头
# 设置长度
.SET HEADER_LENGTH, multiboot_header_end - multiboot_header
# 设置校验和
.SET CHECKSUM, -(MULTIBOOT2_HEADER_MAGIC + MULTIBOOT_ARCHITECTURE_I386 +
HEADER_LENGTH)
# 要求按照 MULTIBOOT_HEADER_ALIGN 对齐
.align MULTIBOOT_HEADER_ALIGN
# 指定所在 section
.section .multiboot_header
multiboot_header:
# 按照 Multiboot2 规范，文件头格式如下
    # 魔数
    .long MULTIBOOT2_HEADER_MAGIC
    # 架构
    .long MULTIBOOT_ARCHITECTURE_I386
    # 长度
    .long HEADER_LENGTH
    # 校验和
    .long CHECKSUM
    # 其它
    # 添加其它内容在此，详细信息见 Multiboot2 Specification version 2.0.pdf
    # 结束标记
    .short MULTIBOOT_HEADER_TAG_END
    .short 0
    .long 8
multiboot_header_end:

# 指定 test section
.section .text
# 声明函数
.global _start
# 声明外部函数
.extern kernel_main
.type _start, @function
_start:
    # 关中断
    cli
    # 设置栈地址
    mov $STACK_TOP, %esp
    # 栈地址按照 16 字节对齐
    and $0xFFFFFFF0, %esp
    # 帧指针修改为 0
    mov $0, %ebp
    push $0
    popf
    # multiboot2_info 结构体指针
```

```
    push %ebx
    # 魔数
    push %eax
    call kernel_main
1:  hlt
    jmp 1b
    ret
    ret

# 预留栈空间
.section .bss
STACK:
    .skip 16384
STACK_TOP:
```