

OS lab3 report

Name: 刘乐奇

sid: 12011327

Ubuntu用户名: lynchrocket

请详细描述本节课最小化内核的启动过程

执行 `make qemu` 时，对应于执行

```
qemu-system-riscv64 \  
-machine virt \  
-nographic \  
-bios default \  
-device loader,file=bin/ucore.bin,addr=0x80200000
```

这条指令相当于给模拟的计算机插电，然后qemu会调用内置的OpenSBI作为我们的bootloader。OpenSBI 所做的一件事情就是把 CPU 从 M Mode 切换到 S Mode，接着跳转到一个固定地址 0x80200000，开始执行内核代码。

ELF和BIN文件的区别是什么

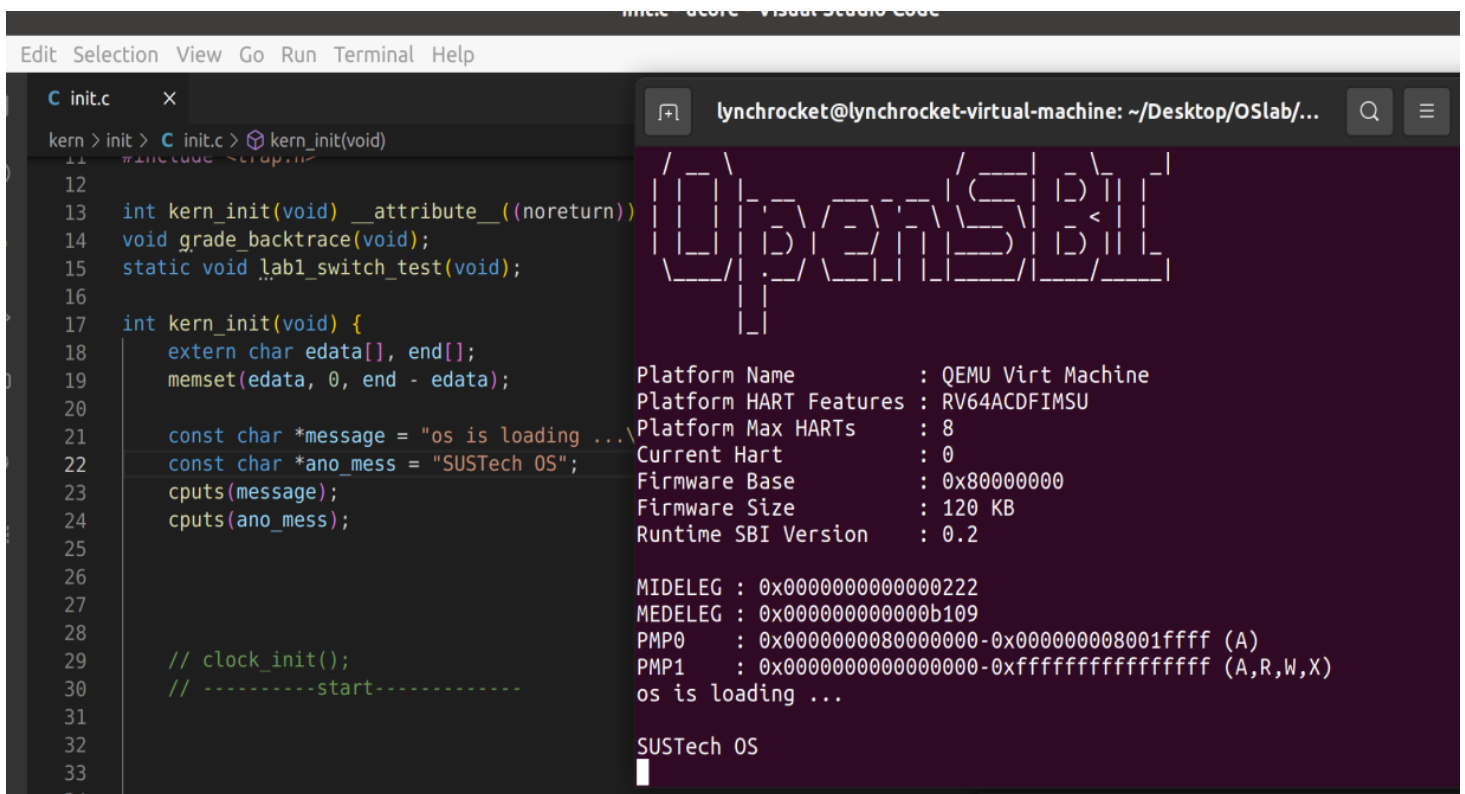
ELF文件是linux系统上的主要可执行文件的格式，比较复杂；BIN文件是二进制可执行文件，比较简单。

链接脚本的作用是什么

链接器能把输入文件（往往是.o文件）连接成输出文件（往往是elf文件），而输入文件和输出文件都有很多section。链接脚本会描述怎样把输入文件的section映射到输出文件的section, 同时规定这些section的内存布局。

在init.c（截图）使用cputs函数，使得在最小化内核启动后通过cputs打印出“SUSTech OS”（截图）

init.c代码如截图左边，cputs打印效果如截图右边。



```
kern > init > C init.c > kern_init(void)
11 #include <stdio.h>
12
13 int kern_init(void) __attribute__((noreturn));
14 void grade_backtrace(void);
15 static void lab1_switch_test(void);
16
17 int kern_init(void) {
18     extern char edata[], end[];
19     memset(edata, 0, end - edata);
20
21     const char *message = "os is loading ...";
22     const char *ano_mess = "SUSTech OS";
23     cputs(message);
24     cputs(ano_mess);
25
26
27
28
29     // clock_init();
30     // -----start-----
31
32
33
34
```

```
Platform Name      : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs   : 8
Current Hart        : 0
Firmware Base       : 0x80000000
Firmware Size       : 120 KB
Runtime SBI Version  : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffff (A,R,W,X)
os is loading ...

SUSTech OS
```

在stdio.c中参考cputs()函数实现double_puts()函数（截图），将输出的字符串每个字符打印两次，如double_puts("SUSTech")应输出"SSUUSSTTeecchh"。在init.c中调用该函数（截图），并使得最小化内核启动后输出“IILLOOVVEEOOSS”（截图）。

在 kern/libs/stdio.c 中编写如下代码

```
C init.c      C stdio.h      C stdio.c 2 X
kern > libs > C stdio.c > ...
49  /* *
50  * cputs- writes the string pointed by @str
51  * appends a newline character.
52  * */
53  int cputs(const char *str) {
54      int cnt = 0;
55      char c;
56      while ((c = *str++) != '\0') {
57          cputch(c, &cnt);
58      }
59      cputch('\n', &cnt);
60      return cnt;
61  }
62
63  int double_puts(const char *str) {
64      int cnt = 0;
65      char c;
66      while((c = *str++) != '\0') {
67          cputch(c, &cnt);
68          cputch(c, &cnt);
69      }
70      cputch('\n', &cnt);
71      return cnt;
72  }
73
```

在 libs/stdio.h 中增加函数声明

C init.c

C stdio.h

X

C stdio.c 2

libs > C stdio.h > double_puts(const char *)

```
1  #ifndef __LIBS_STDIO_H__
2  #define __LIBS_STDIO_H__
3
4  #include <defs.h>
5  #include <stdarg.h>
6
7  /* kern/libs/stdio.c */
8  int cprintf(const char *fmt, ...);
9  int vcprintf(const char *fmt, va_list ap);
10 void cputchar(int c);
11 int cputs(const char *str);
12 int double_puts(const char *str);
13 int getchar(void);
14
15 /* kern/libs/readline.c */
16 char *readline(const char *prompt);
17
18 /* libs/printfmt.c */
19 void printfmt(void (*putch)(int, void *), void *putdat,
20 void vprintfmt(void (*putch)(int, void *), void *putdat,
21 int snprintf(char *str, size_t size, const char *fmt,
22 int vsnprintf(char *str, size_t size, const char *fmt,
23
24 #endif /* !__LIBS_STDIO_H__ */
25
```

在 kern/init/init.c 中编写如下代码

