

CS302 OS Lab03 - Report

Name: 刘仁杰

SID: 11911808

Answers

1.请详细描述本节课最小化内核的启动过程

- 把entry.S和init.c, stdio.c等几个c文件编译成为.o目标文件
- 链接器将.o文件链接成可执行文件 (elf文件)
- objcopy把elf文件转化成为ucore.bin文件, 即最小化内核的硬盘
- 在QEMU模拟的riscv计算机里, OpenSBI固件充当bootloader, 此时计算机运行在M态, OpenSBI访问最小化内核的硬盘 (ucore.bin文件), 将二进制可执行文件读取到内存的特定位置, 然后将program counter跳转到内存地址为0x80200000的位置, 并且将计算机切换到S态, program counter开始读取相应内存地址的指令, 开始执行内核代码。

2.ELF和BIN文件的区别是什么

1. ELF :

包含一个文件头(ELF header), 包含冗余的调试信息, 指定程序每个section的内存布局, 需要解析program header才能知道各段(section)的信息, 每个section的被加载到哪个内存地址块是不确定的。

2. BIN :

没有多余的信息, 简单地在文件头之后解释自己应该被加载到什么起始位置, OpenSBI可以很好的处理。

3.链接脚本的作用是什么

把输入文件(往往是.o文件)链接成输出文件(往往是elf文件), 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。

4.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)

```
File Edit Selection View Go Run Terminal Help

EXPLORER
LAB3
  .vscode
  bin
  kern
    driver
    init
      entry.S
      init.c
    libs
      stdio.c
    mm
      memlayout.h
      mmu.h
    libs
      defs.h
      error.h
      printfmt.c
      readline.c
      riscv.h
      sbi.c
      sbi.h
      stdarg.h
      stdio.h
      string.c
      string.h
  obj
  tools
  Makefile

C init.c  u x  C stdio.h  u  C stdio.c  u

kern > init > C init.c > kern_init(void)
1  #include <stdio.h>
2  #include <string.h>
3  #include <console.h>
4
5  int kern_init(void) __attribute__((noreturn));
6
7  int kern_init(void) {
8      extern char edata[], end[];
9      memset(edata, 0, end - edata);
10
11      const char *message = "os is loading ...\n";
12      cputs(message);
13
14      const char* msg = "SUSTech OS\n";
15      cputs(msg);
16
17      const char* double_msg = "ILOVEOS\n";
18      double_puts(double_msg);
19
20      while (1)
21          ;
22  }
23
```

```
lrj11911808@lrj-Precision-3630-Tower: ~/2022_os/lab/lab03/code_lab3/lab3
lrj11911808@lrj-Precision-3630-Tower:~/2022_os/lab/lab03/code_lab3/lab3$ make qemu
+ cc kern/init/init.c
+ cc kern/libs/stdio.c
+ cc libs/printfmt.c
+ cc libs/readline.c
+ ld bin/kernel
riscv64-unknown-elf-objcopy bin/kernel --strip-all -O binary bin/ucore.bin

OpensBI v0.6

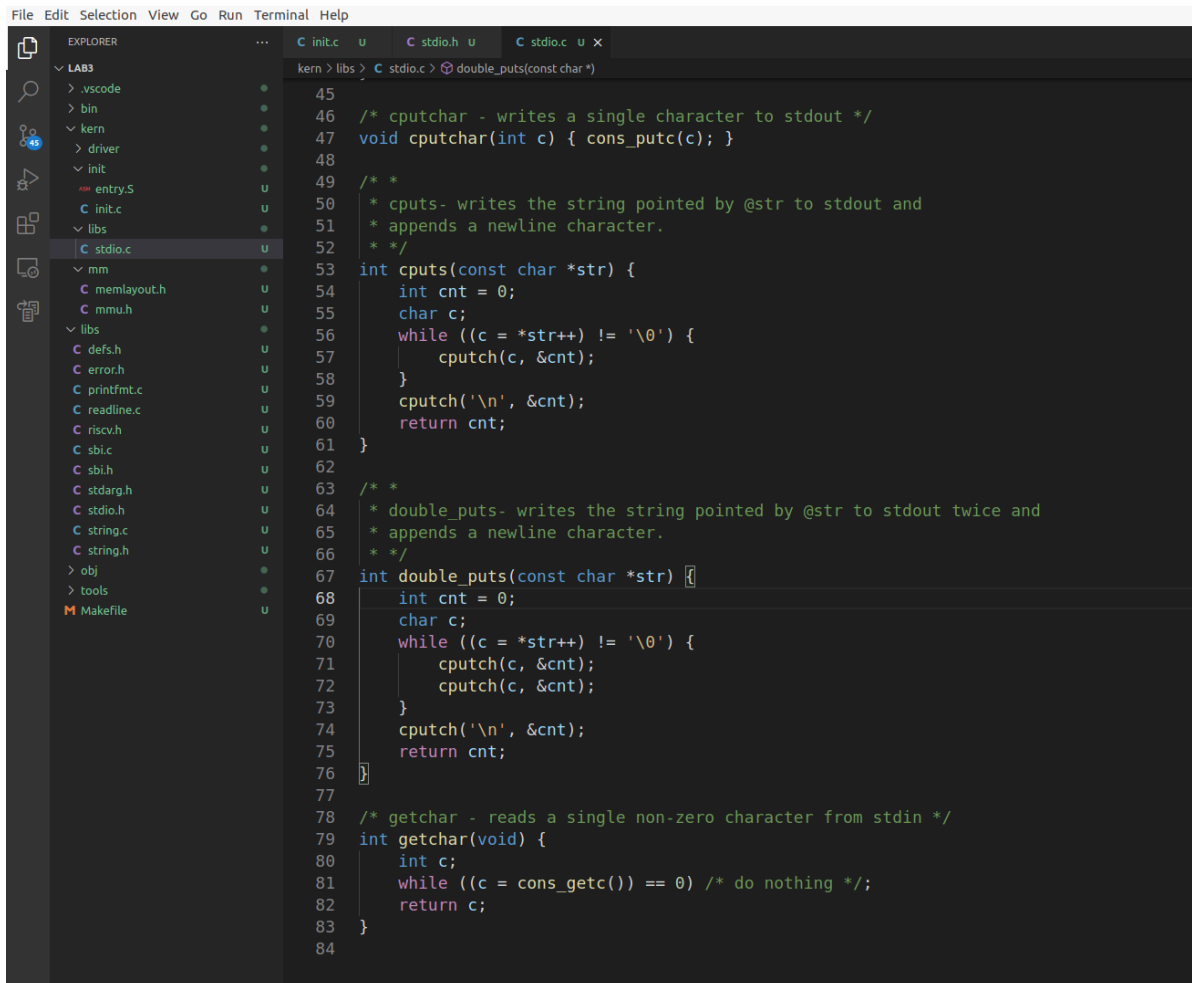
Platform Name       : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs   : 8
Current Hart        : 0
Firmware Base       : 0x80000000
Firmware Size       : 120 KB
Runtime SBI Version  : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000000000000-0x000000000001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffff (A,R,W,X)
os is loading ...

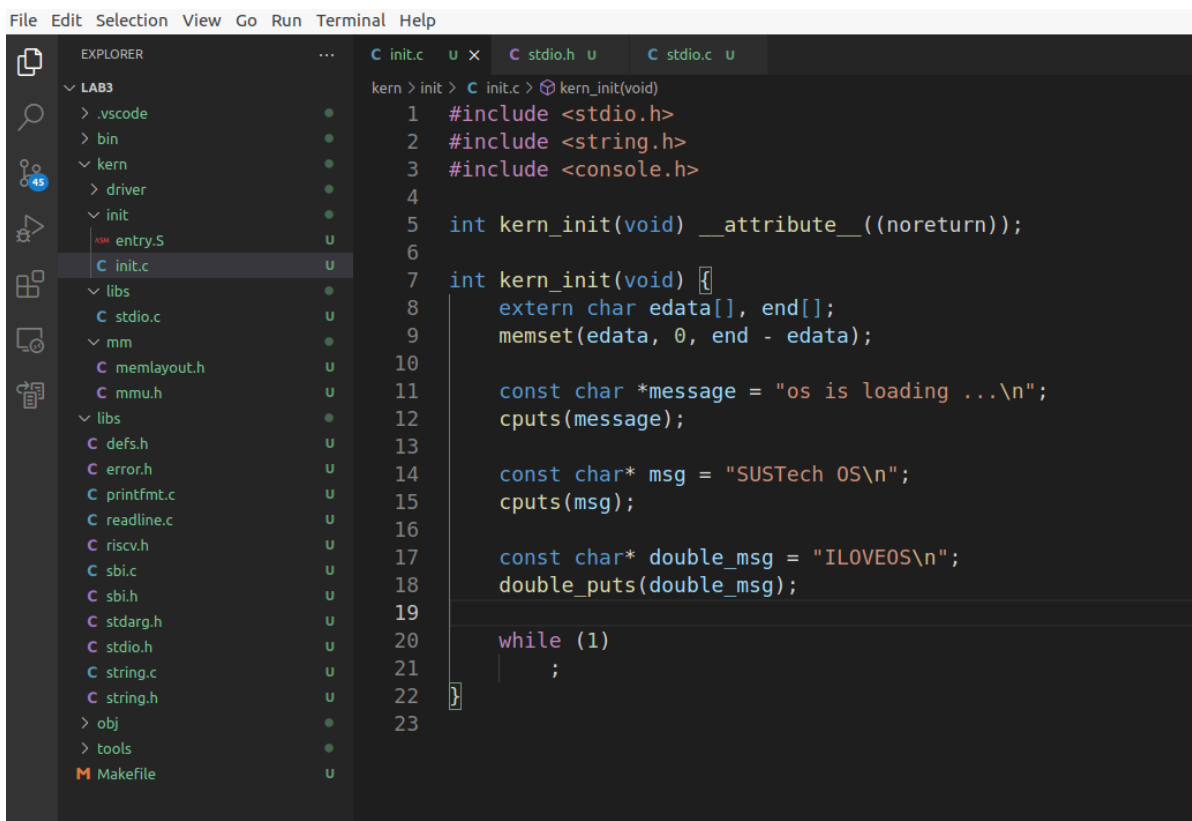
SUSTech OS

IILL00VVEE00SS
```

5.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）



```
File Edit Selection View Go Run Terminal Help
EXPLORER
LAB3
  .vscode
  bin
  kern
  driver
  init
  entry.S
  init.c
  libs
  stdio.c
  mm
  memlayout.h
  mmu.h
  libs
  defs.h
  error.h
  printfmt.c
  readline.c
  riscv.h
  sbi.c
  sbi.h
  stdarg.h
  stdio.h
  string.c
  string.h
  obj
  tools
  Makefile
C init.c
C stdio.h
C stdio.c
kern > libs > C stdio.c > double_puts(const char *)
45
46 /* cputc - writes a single character to stdout */
47 void cputc(int c) { cons_putc(c); }
48
49 /* *
50 * cputs- writes the string pointed by @str to stdout and
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         cputc(c, &cnt);
58     }
59     cputc('\n', &cnt);
60     return cnt;
61 }
62
63 /* *
64 * double_puts- writes the string pointed by @str to stdout twice and
65 * appends a newline character.
66 * */
67 int double_puts(const char *str) {
68     int cnt = 0;
69     char c;
70     while ((c = *str++) != '\0') {
71         cputc(c, &cnt);
72         cputc(c, &cnt);
73     }
74     cputc('\n', &cnt);
75     return cnt;
76 }
77
78 /* getchar - reads a single non-zero character from stdin */
79 int getchar(void) {
80     int c;
81     while ((c = cons_getc()) == 0) /* do nothing */;
82     return c;
83 }
84
```



```
File Edit Selection View Go Run Terminal Help
EXPLORER
LAB3
  .vscode
  bin
  kern
  driver
  init
  entry.S
  init.c
  libs
  stdio.c
  mm
  memlayout.h
  mmu.h
  libs
  defs.h
  error.h
  printfmt.c
  readline.c
  riscv.h
  sbi.c
  sbi.h
  stdarg.h
  stdio.h
  string.c
  string.h
  obj
  tools
  Makefile
C init.c
C stdio.h
C stdio.c
kern > init > C init.c > kern_init(void)
1 #include <stdio.h>
2 #include <string.h>
3 #include <console.h>
4
5 int kern_init(void) __attribute__((noreturn));
6
7 int kern_init(void) {
8     extern char edata[], end[];
9     memset(edata, 0, end - edata);
10
11     const char *message = "os is loading ...\n";
12     cputs(message);
13
14     const char* msg = "SUSTech OS\n";
15     cputs(msg);
16
17     const char* double_msg = "ILOVEOS\n";
18     double_puts(double_msg);
19
20     while (1)
21     ;
22 }
23
```

```
lrj11911808@lrj-Precision-3630-Tower: ~/2022_os/lab/lab03/code_lab3/lab3
lrj11911808@lrj-Precision-3630-Tower:~/2022_os/lab/lab03/code_lab3/lab3$ make qemu
+ cc kern/init/init.c
+ cc kern/libs/stdio.c
+ cc libs/printfmt.c
+ cc libs/readline.c
+ ld bin/kernel
riscv64-unknown-elf-objcopy bin/kernel --strip-all -O binary bin/ucore.bin

OpenSBI v0.6

      _ _ _ _ _
     / / / / /
    / / / / /
   / / / / /
  / / / / /
 / / / / /
/_/_/_/_/_

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    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
os is loading ...

SUSTech OS

IILL00VVEE00SS
```