Week3 Report in Class (Fri56)

11911839 聂雨荷

Q1

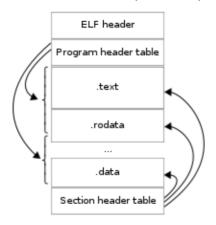
请详细描述本节课最小化内核的启动过程

- make: 生成kernel的可执行文件 ucore.bin 文件,作为我们的硬盘
 - 。 把entry.S和init.c, stdio.c等几个c文件编译成为.o目标文件, 然后链接器[3]将.o文件链 接成可执行文件kernel (elf文件), 最后使用objcopy把elf文件转化成为ucore.bin[2], 这是装有我们最小化操作系统内核的二进制文件
- make qemu: 用qemu中自带的OpenSB作为的 bootloader, 启动我们的内核
 - 。 给的模拟计算机插电
 - 。 gemu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

Q2

ELF和BIN文件的区别是什么

• elf文件包含一个文件头(ELF header)和冗余的调试信息,指定程序每个section的内存布局



• bin文件会存储所有的数据信息,它会把elf文件指定的每段的内存布局都映射到一块线性的数据里,这块线性的数据(或者说程序)加载到内存里就符合elf文件之前指定的布局

链接脚本的作用是什么

一般来说,输入文件和输出文件都有很多section,链接脚本(linker script)的作用,就是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。

Q4

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)

```
kern > init > C init.c > 分 kern_init(void)
  1 #include <clock.h>
     #include <console.h>
  3
    #include <defs.h>
    #include <intr.h>
  4
    #include <kdebug.h>
  5
  6 #include <kmonitor.h>
    #include <pmm.h>
  7
  8 #include <riscv.h>
  9 #include <stdio.h>
 10 #include <string.h>
 #include <trap.h>
 12
    int kern init(void) attribute ((noreturn));
 13
     void grade backtrace(void);
 14
     static void lab1_switch_test(void);
 15
 16
     int kern init(void) {
 17
         extern char edata[], end[];
 18
         memset(edata, 0, end - edata);
 19
 20
        const char *message = "os is loading ...\n";
 21
         cputs(message);
 22
 23
         // clock_init();
 24
         // -----start-----
 25
         const char *sustech_msg = "SUSTech OS\n";
 26
 27
         cputs(sustech msg);
 28
         // -----end-----
 29
 30
         while (1)
 31
 32
           ;
 33
 34
```

```
nyh11911839@nyh-virtual-machine: ~/OSlab/lab3/lab
                                   i l') li
Platform Name
                         : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs
Current Hart
                         : 0
Firmware Base
                         : 0x80000000
Firmware Size
                         : 120 KB
Runtime SBI Version
MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMPO : 0x000000080000000-0x00000008001ffff (A)
PMP1 : 0x00000000000000-0xffffffffffffff (A,R,W,X)
os is loading ...
SUSTech OS
```

Q5

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"IIDDOONNTTLLOOVVEEOOSS"(截图)。

```
/*
* [Lab-3] Question 5
* Implement double_put() function, output each char twice
* eg: double_puts("SUSTech") -> "SSUUSSTTeecchh"
*/
int double_puts(const char *str){
   int cnt = 0;
   char c;
   while ((c = *str++) != '\0') {
      cputchar(c);
      cputch(c, &cnt);
   }
   cputch('\n', &cnt);
   return cnt;
}
```

```
C init.c
lab > kern > init > C init.c > 🛇 kern_init(void)
  6 #include <kmonitor.h>
     #include <pmm.h>
      #include <riscv.h>
      #include <stdio.h>
  9
 10
     #include <string.h>
      #include <trap.h>
 11
 12
 13
      int kern_init(void) __attribute__((noreturn));
      void grade_backtrace(void);
 14
 15
      static void lab1_switch_test(void);
 16
      int kern_init(void) {
 17
 18
          extern char edata[], end[];
          memset(edata, 0, end - edata);
 19
 20
          const char *message = "os is loading ...\n";
 21
 22
          cputs(message);
 23
          // clock_init();
 24
 25
          // -----start-----
          const char *love msg = "IDONTLOVEOS";
 26
          double_puts(love_msg);
 27
 28
          // ----end-----
 29
 30
 31
          while (1)
 32
          ;
 33
```

```
Ħ
                 nyh11911839@nyh-virtual-machine: ~/OSlab/lab3/lab
Platform Name
                           : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs
Current Hart
                           : 0
Firmware Base
                           : 0x80000000
Firmware Size
Runtime SBI Version
                           : 120 KB
                           : 0.2
MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
        : 0x0000000080000000-0x00000008001ffff (A)
: 0x000000000000000-0xffffffffffffff (A,R,W,X)
PMP0
PMP1
os is loading ...
IIDDOONNTTLLOOVVEEOOSS
```