Lab3 Report

12011702 张镇涛

1

1.使用make 命令生成我们的模拟硬盘

我们在项目目录下面执行make命令，在bin目录下会生成ucore.bin文件，这是我们的硬盘。

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。接下来我们使用qemu中自带的OpenSBI[1]作为我们的 bootloader，启动我们的内核。

2.使用make qemu 启动我们的内核

相当于给我们的模拟计算机插电，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。

2

ELF文件是linux系统上的主要可执行文件的格式(图片来自wiki)。BIN文件是二进制可执行文件。

bss段是一个初始化为零的一个大数组，在elf文件里是 bss数据段的一部分，只需要记住这个数组的起点和终点就可以了，等到加载到内存里的时候分配 那一段内存。但是在bin文件里，那个数组有多大，有多少个字节的0，bin文件就要对应有多少个零。

实际上，可以认为bin文件会把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。

3

一般来说，输入文件（往往是.o文件）和输出文件（往往是elf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。

4

```
os12011702@vmos-tony:~/oslab/lab3/lab$ make qemu

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)
SUSTech OS
```

Open | init.c ~/oslab/lab3/lab/k... | Save

```
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 = "SUSTech OS";
22     cputs(message);
23
24
25
26
27     // clock_init();
28     // ----------start-------------
29
30
31
```

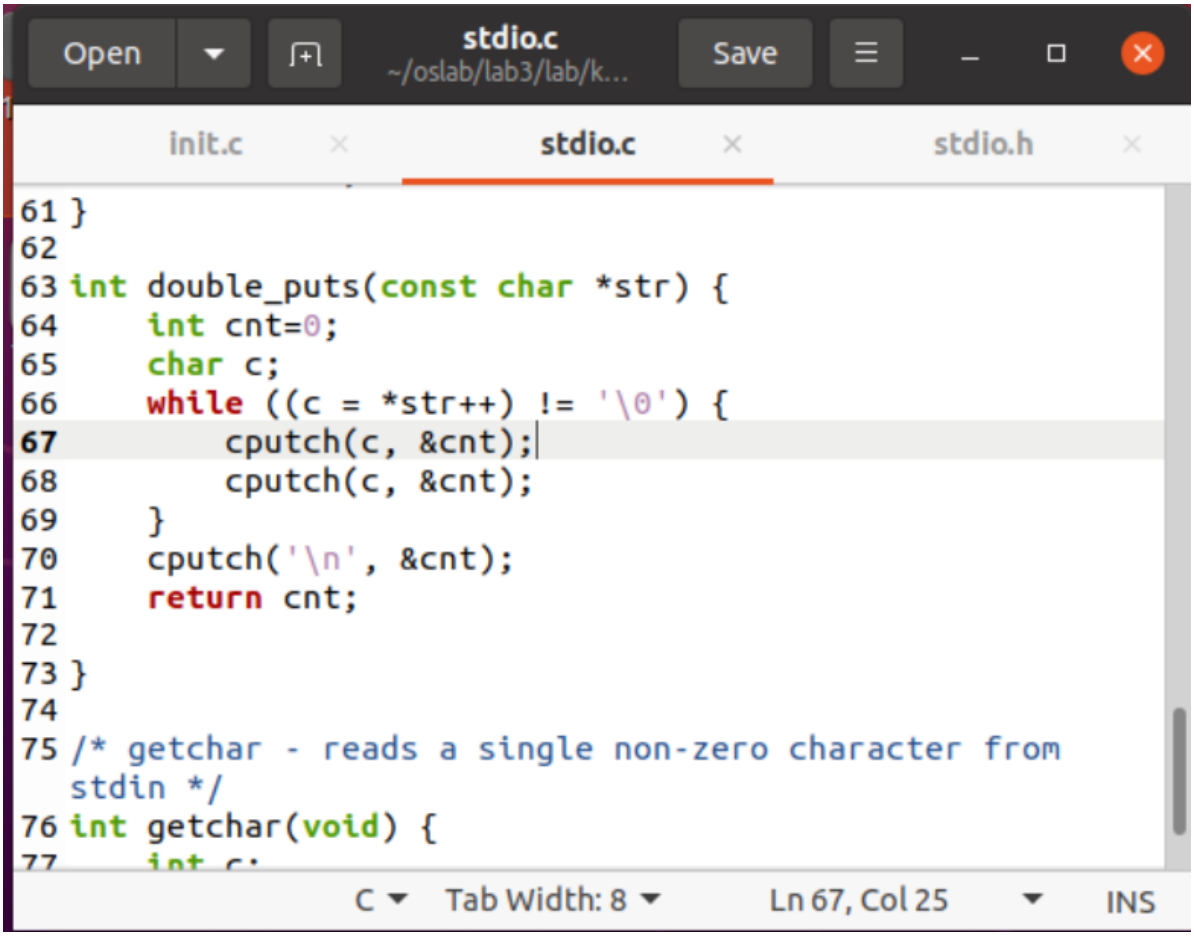
C | Tab Width: 8 | Ln 21, Col 38 | INS

```
os12011702@vmos-tony:~/oslab/lab3/lab$ make qemu
+ cc kern/init/init.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)
IIDDOONNTTLLOOVVEEOOSS
```

Open | init.c ~/oslab/lab3/lab/k... | Save

```c
#include <string.h>
#include <trap.h>

int kern_init(void) __attribute__((noreturn));
void grade_backtrace(void);
static void lab1_switch_test(void);

int kern_init(void) {
    extern char edata[], end[];
    memset(edata, 0, end - edata);

    const char *message = "IDONTLOVEOS";
    double_puts(message);
    //cputs(message);



    // clock_init();
    // ----------start-------------
```

C | Tab Width: 8 | Ln 24, Col 1 | INS


Open
stdio.c
~/oslab/lab3/lab/k...
Save
init.c
stdio.c
stdio.h
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 }
74
75 /* getchar - reads a single non-zero character from stdin */
76 int getchar(void) {
C
Tab Width: 8
Ln 67, Col 25
INS