Project1 Bootloader 设计文档

中国科学院大学
李静逸
2017.9.27

1. Bootblock 设计流程

（1）Bootblock 主要完成的功能

Bootblock 是程序运行最开始执行的程序，需要从启动设备读取内核，加载到内存中。

（2）Bootblock 被载入内存后的执行流程

将 kernel 在内存的起止地址、main 函数偏移量、文件长度传给读盘函数以读取 kernel，然后开始执行 kernel。

（3）Bootblock 如何调用 SD 卡读取函数

跳转到读取函数的起始地址 0x8007b1a8.

（4）Bootblock 如何跳转至 kernel 入口

跳转到 kernel_ehdr->e_entry

（5）任何在设计、开发和调试 bootblock 时遇到的问题和解决方法

最开始找不到 kernel 的入口地址，后来用 objdump –f kernel 命令看到了 kernel 入口为 0xa080026c

2. Createimage 设计流程

（1）Bootblock 编译后的二进制文件、Kernel 编译后的二进制文件，以及 SD 卡 image 文件这三者之间的关系

Image 文件第一个扇区存的是 bootblock 编译后的二进制文件，从第二个扇区开始存的是 kernel 编译后的二进制文件。

（2）如何获得 Bootblock 和 Kernel 二进制文件中可执行代码的位置和大小

通过 program header 中的 p_filesz 获得二进制文件中可执行代码的大小。

通过 program header 中的 p_offset 获得二进制文件中可执行代码的偏移量，偏移量加上文件起始地址就是可执行代码的位置。

（3）如何让 Bootblock 获取到 Kernel 二进制文件的大小，以便进行读取

通过 program header 中的 p_filesz 得到 kernel 二进制文件的大小。由于 bootblock 在 image 的第一个扇区，所以 bootblock 中给读取函数第三个参数赋值的那一行命令的地址相对 bootblock 起始地址的偏移量，就是 image 中相应代码相对 image 起始地址的偏移量。通过命令 mipsel –linux objdump –d bootblock 得到偏移量。利用 fseek 函数定位到相应位置，修改读取函数的第三个参数为 kernel 二进制文件的大小，即 kernel_phdr->p_filesz。

（4）任何在设计、开发和调试 createimage 时遇到的问题和解决方法

在最开始运行 createimage 时，报出 segmentation fault，用 gdb 调试发现是 kernel 函数的 ELF header 和 program header 地址有问题，得到的数据不对，由于 read_exec_file()这个函数实在是很麻烦，让指针转了好几道弯，很容易出错，于是我删掉了 read_exec_func()函数，直接在 main 函数里实现相应功能，程序成功运行。

在输入 make 命令时出现如下图所示报错

```
stu@ubuntu:~/task_3_start_code/task_3_start_code$ make
mipsel-linux-gcc  -G 0 -O2  -fno-pic -mno-abicalls -fno-builtin -nostdinc -mips3 -Ttext=0xffffffffa0800000 -N -o bootblock bootblock.s -nostdlib -e main -Wl,-m -Wl,elf32lt
smip -T ld.script
mipsel-linux-gcc  -G 0 -O2  -fno-pic -mno-abicalls -fno-builtin -nostdinc -mips3 -Ttext=0xffffffffa0800200 -N -o kernel kernel.c -nostdlib -e main -Wl,-m -Wl,elf32ltsmip -
T ld.script
mipsel-linux-gcc      createimage.c   -o createimage
/opt/gcc-3.4.6-2f/bin/../lib/gcc/mipsel-linux/3.4.6/../../../../mipsel-linux/bin/ld: cannot find /home/cpu/gcc-3.4.6-2f/mipsel-linux//lib/libc.so.6
collect2: ld returned 1 exit status
<builtin>: recipe for target 'createimage' failed
make: *** [createimage] Error 1
```

后来发现是因为 Makefile 文件没有更改。

3. 关键函数功能

```
void write_kernel(FILE **image_file, FILE *kernel_file, Elf32_Ehdr *kernel_ehdr, Elf32_Phdr
*kernel_phdr) //将可执行文件kernel写入内核镜像image文件中
{
    Elf32_Word size = kernel_phdr->p_filesz;
    uint8_t buf[size];
    fseek(kernel_file, kernel_phdr->p_offset, SEEK_SET);
    fread(buf, size, 1, kernel_file);
    fseek(*image_file, 512, SEEK_SET);
    fwrite(buf, 1, size, *image_file);
}

void record_kernel_sectors(FILE **image_file, Elf32_Ehdr *kernel_ehdr, Elf32_Phdr
*kernel_phdr, int num_sec)   //将kernel的扇区个数写入bootblock的os_size位置处
{
    fseek(*image_file, 0x40, SEEK_SET);
    int inst = 0x34060000 | kernel_phdr->p_filesz;
    fwrite(&inst, 4, 1, *image_file);
}



int main(int argc, char *argv[])
{
   //main函数反映了整个程序的思路
    FILE *boot_file = NULL, *kernel_file = NULL, *image_file = NULL;
    Elf32_Phdr *boot_phdr = NULL, *kernel_phdr = NULL;
    Elf32_Ehdr *boot_ehdr = NULL, *kernel_ehdr = NULL;
    int num_sec = 0, k_phnum = 0;

    image_file = fopen("image", "w");

    //read kernel and bootbloc
      uint8_t boot_buf[4096];
      boot_file = fopen("bootblock", "rb");
      fread(boot_buf, 4096, 1, boot_file);
      boot_ehdr = (void*) boot_buf;
```

```
    boot_phdr = (Elf32_Phdr*)((void*)boot_buf + boot_ehdr->e_phoff);
    uint8_t kernel_buf[4096];
    kernel_file = fopen("kernel", "rb");
    fread(kernel_buf, 4096, 1, kernel_file);
    kernel_ehdr = (void*) kernel_buf;
    kernel_phdr = (Elf32_Phdr*)((void*)kernel_buf + kernel_ehdr->e_phoff);

  //write boot and kernel to image
  write_bootblock(&image_file, boot_file, boot_ehdr, boot_phdr);
  write_kernel(&image_file, kernel_file, kernel_ehdr, kernel_phdr);

  //change bootblock
  num_sec = (kernel_phdr->p_filesz)/0x200 + 1;
  record_kernel_sectors(&image_file, kernel_ehdr, kernel_phdr, num_sec);

  //printf info
  k_phnum = kernel_ehdr->e_phnum;
  extended_opt(boot_phdr, k_phnum, kernel_phdr, num_sec);

  fclose(boot_file);
  fclose(kernel_file);
  fclose(image_file);
  return 0;
}
```

参考文献

[1] http://wiki.jikexueyuan.com/project/c/
[2] https://www.ibm.com/developerworks/cn/linux/l-excutff/

■