操作系统lab1实验报告

2012011350 计24 周琳钧

练习1：

1. 10~23行为GCC的匹配，保存值为GCCPREFIX，26~38行为qemu的匹配，保存值为QEMU，46~89行定义了一些常用命令和工具，比如GDB和objdump等等，分别用特定的字符串保存，95~110行定义了输出文件， 从113行开始为makefile的主程序，147~172行用到kernel.ld生成了基础内核映像kernel，用到的文件是kern下的一些源文件编译成的.o文件, 177~189行生成了bootblock，用到三个文件共同编译：bootasm.o（源自bootasm.S），bootmain.o（源自bootmain.c），sign，194~195行生成了tools/sign，200~209行生成了最终文件ucore.img,生成方法是生成10000个块，每块512字节，然后把bootblock写到第一个块中，然后从第二个块中开始写入内容。

重要的命令有：

1. gcc -Iboot/ -fno-builtin -Wall -ggdb -m32 –gstabs -nostdinc -fno-stack-protector -Ilibs/ -Os -nostdinc -c boot/bootasm.S -o obj/boot/bootasm.o

-ggdb 生成可供gdb使用的调试信息

-m32 生成适用于32位环境的代码

-gstabs 生成stabs格式的调试信息

-nostdinc 不使用标准库

-fno-stack-protector 不生成用于检测缓冲区溢出的代码

-Os 为减小代码大小而进行优化

-I<dir> 添加搜索头文件的路径

1. ld -m elf\_i386 -nostdlib -N -e start -Ttext 0x7C00 obj/boot/bootasm.o obj/boot/bootmain.o -o obj/bootblock.o

-m <emulation> 模拟为i386上的连接器

-nostdlib 不使用标准库

-N 设置代码段和数据段均可读写

-e <entry> 指定入口

-Ttext 制定代码段开始位置

1. objcopy -S -O binary obj/bootblock.o obj/bootblock.out

-S 移除所有符号和重定位信息

-O <bfdname> 指定输出格式

1. 一个合理的主引导扇区由4部分组成：
2. 引导程序：446字节
3. 分区表：64字节
4. 结束标志：55AA，2字节

练习2：

1. 运行make debug可以启动gdb
2. b \*7C00即可
3. 几乎完全相同
4. 略

练习3：

1. A20含义：早期8086机器中只有20根地址线，内存访问最大为1MB，当访问1M以上的地址时会发生回卷现象，而80386机器上不会产生这种现象，因为它的地址线大于20根，为保证兼容性，设置了一个A20开关，操作系统加载时A20是禁止的，所以我们要先激活A20:

打开

in al,92h ;读Fast A20的端口

or al,02

out 92h,al ;A20的bit 1为1时，开启A20

关闭：

in al,92h ;读Fast A20的端口

and al,0fdh

out 92h,al ;A20的bit 1为0时，关闭A20

初始化GDT表：GDT（全局描述符表）的地址被存放在了GDTR寄存器中，访问GDTR可以加载GDT，GDT中存放的是各个进程LDT表的位置信息。

初始化GDT表可用 lgdt gdtdesc解决

保护模式切换：修改cr0寄存器上PE位置为1，并通过长跳转的方式更新cs寄存器的基地址。

最后建立堆栈，程序结束。

练习4：

1. 本练习将两个实验综合回答：

主程序存放在bootmain这个函数中：

第一步：获取标识

读取硬盘中第1个扇区并找到其标识符表明是否是一个正常的ELF文件

第二步：读取硬盘数据

主要是readsect()这个函数，它先等待硬盘空闲，若空闲，则开始读取ELF格式文件。读取完成后，找到其内核的入口地点。

练习5：

1. ebp为栈基址，eip是代码地址，args是可能参数
2. 程序见源码

练习6：

程序见源码。

1. 8字节，其中2-3字节是段选择子，0-1字节和6-7字节拼成位移，两者联合便是中断处理程序的入口地址