实验 1： Multiboot 启动

目录

原理说明

- Multiboot Header 格式约定

  Multiboot Hearder的文件头至少要有12个字节的检验核，以便Qemu认出。

  其格式如下表：

Offset	Type	Field Name	Note
0	u32	magic	必要项，应为 0x1BADB002
4	u32	flags	必要项，可以是 0
8	u32	checksum	必要项，应满足 magic + checksum + flags = 0
...	...	...	...

- VGA输出 本实验通过直接修改VGA显存来显示字符串。

  已知VGA显存的起始地址是　0xB8000，因此从该地址开始写入要显示的文本。

  VGA显存约定每个字符需要两个字节，一个字节用于存放ASCII码，另外一个用于存放该字符的显示样式。具体格式如下表：

Attribute	Charactor
15~8	7~0

  其中字符样式Attribute还有如下格式：

Blink	Background Color	Foreground Color
7	6~4	3~0

  通过汇编代码 `movl $0x2f4b2f4f, 0xB8000` 可以直接修改显存，输出绿底白字的 "OK"。

- 串口输出

在Qemu中，不初始化串口波特率等参数，也能正确输出。串口的端口地址是 0x3F8，调用 outb() 并正确指定其参数即可在串口输出字符信息。

outb() 的定义： `void outb (unsigned char data, unsigned short port);`

源代码说明

- multiboot_header

```
.section .multiboot_header
.long 0x1BADB002        /* magic        */
.long 0x0               /* flags        */
.long -0x1BADB002       /* checksum     */
```

- uart输出 "helloworld"

```
.section .text
start:
    movw $0x3f8, %dx
    nop
    nop

    movb $0x68, %al         /* putchar h    */
    outb %al, %dx
    movb $0x65, %al         /* putchar e    */
    outb %al, %dx
    movb $0x6c, %al         /* putchar l    */
    outb %al, %dx
    movb $0x6c, %al         /* putchar l    */
    outb %al, %dx
    movb $0x6f, %al         /* putchar o    */
    outb %al, %dx
    movb $0x77, %al         /* putchar w    */
    outb %al, %dx
    movb $0x6f, %al         /* putchar o    */
    outb %al, %dx
    movb $0x72, %al         /* putchar r    */
    outb %al, %dx
    movb $0x6c, %al         /* putchar l    */
    outb %al, %dx
    movb $0x64, %al         /* putchar d    */
    outb %al, %dx
```

- VGA输出 "hellworld, <姓名>_<学号>"

```
/* 以4字节为单位，修改VGA显存，输出指定字符串 */
helloworld:
    movl $0x2f652f68, 0xB8000
```

```
    movl $0x2f6c2f6c, 0xB8004
    movl $0x2f772f6f, 0xB8008
    movl $0x2f722f6f, 0xB800C
    movl $0x2f642f6c, 0xB8010
    movl $0x2f202f2c, 0xB8014
    movl $0x2f422f50, 0xB8018
    movl $0x2f382f31, 0xB801C
    movl $0x2f312f31, 0xB8020
    movl $0x2f362f31, 0xB8024
    movl $0x2f302f38, 0xB8028
    movl $0x2f472f5f, 0xB802C
    movl $0x2f6f2f75, 0xB8030
    movl $0x2f652f57, 0xB8034
    movl $0x2f642f69, 0xB8038
    movl $0x2f6e2f6f, 0xB803C
    movl $0x00002f67, 0xB8040
```

- VGA输出清除界面上的 Qemu 自带提示信息

```
/* 循环擦除一段VGA显存，然后程序暂停 */
clean:
        movl $0, %ebx
loop:
        movl $0xB8044, %eax
        movl $0 , (%eax, %ebx, 4)
        addl $1, %ebx
        cmpl $0x1000, %ebx
        jl loop

        hlt
```

代码布局（地址空间）说明

根据链接描述文件，输出文件的.text从1M位置起始，含有两部分内容。第一部分是输入文件.multiboot_header用于让Qemu识别；按八字节对其后，开始第二部分，第二部分是输入文件的.text即代码部分。

Offset(Base=1M)	Field Name	Note
0	.multiboot_header	multiboot文件头，用于Qemu识别
16	.text	代码块，用于uart、VGA输出

运行和运行结果说明

- 运行

  编译得到 kernel 文件后，在命令行中输入：

  ```
  qemu-system-i386 -kernel multibootHeader.bin -serial stdio
  ```

其中 `-kernel multibootHeader.bin` 指定了目标二进制文件；

`-serial stdio` 的作用是把串口I/O重定向到标准输入输出。

- 运行结果

  Qemu的图形界面被启动，图形界面上输出了预期的字符串信息；Ubuntu命令行界面显示串口的输出信息。

编译过程说明

- Makefile 文件

  将用gcc将multibootHeader.S编译得到multibootHeader.o。

  然后将multibootHeader.o根据multibootHeader.ld链接得到multibootHeader.bin。

  编译链接完成后，删除中间文件multibootHeader.o。

```
ASM_FLAGS= -m32 --pipe -Wall -fasm -g -O1 -fno-stack-protector
multibootHeader.bin: multibootHeader.S
    gcc -c ${ASM_FLAGS} multibootHeader.S -o multibootHeader.o
    ld -n -T multibootHeader.ld multibootHeader.o -o multibootHeader.bin
clean:
    rm -rf ./multibootHeader.bin ./multibootHeader.o
```

- 链接描述文件 multibootHeader.ld

```
/* 输出32位对应i386架构处理器的可执行和可链接文件 */
OUTPUT_FORMAT("elf32-i386", "elf32-i386", "elf32-i386")
/* 指定目标架构为i386 */
OUTPUT_ARCH(i386)
/* 设置程序的入口为start位置 */
ENTRY(start)
/* 定义文件的结构 */
SECTIONS {
    . = 1M;                     /* .text的起始位置为1M */
    .text : {
        *(.multiboot_header)    /* 所有文件的.multiboot_header片段 */
        . = ALIGN(8);           /* 取下一个按8字节对齐的位置 */
        *(.text)                /* 所有文件的.text片段 */
    }
}
```

  根据该链接描述文件链接得到的multibootHeader.bin才能被Qemu识别。

遇到的问题和解决方案

- 问题：WSL 启动 Qemu 后不显示界面

解决方案：下载并安装Xming，在WSL中配置好display环境变量。在Windows下先启动Xming，然后在WSL中启动Qemu就能显示图形界面了。