

《操作系统原理》实验报告

姓名	侯皓斐	学号	U202010851	专业班级	软工 2003 班	时间	2022.04.26
----	-----	----	------------	------	-----------	----	------------

一、实验目的

- 1) 理解操作系统引导程序/BIOS/MBR 的概念和作用;
- 2) 理解并应用操作系统生成的概念和过程;
- 3) 理解并应用操作系统操作界面, 系统调用概念
- 4) 掌握和推广国产操作系统(限银河麒麟或优麒麟, 加 10 分, 直到满分)

二、实验内容

- 1) 用 NASM 编写 MBR 引导程序, 在 BOCHS 虚拟机中测试。
- 2) 在 Linux (建议 Ubuntu 或银河麒麟或优麒麟) 下载并编译 Linux 内核, 并启用新内核。(其他发行版本也可以)
- 3) 为 Linux 内核 (建议 Ubuntu 或银河麒麟或优麒麟) 增加 2 个系统调用, 并启用新的内核, 并编写应用程序测试。(其他发行版本也可以)
- 4) 在 Linux (建议 Ubuntu 或银河麒麟或优麒麟) 或 Windows 下, 编写脚本或批处理。在指定目录中的全部 txt 文件末尾追加或更新: 用户名:日期和时间。root:2022-04-26 21:00

三、实验过程

3.1 编写 MBR 引导程序并测试

开发环境: 银河麒麟操作系统 v10 桌面版, Bochs 与 Nasm。

- 1) 配置环境: 安装 `sudo apt install nasm vgabios bochs bochs-x bximage`
- 2) 编写汇编代码, 用于屏幕显示 “Hello OS”, 获取并显示内存大小信息。此处主要使用了 BIOS 的 INT 10H 中断与 INT 15 号中断。

boot.asm (~/MBR_HelloOS) - 文本编辑器

文件(F) 编辑(E) 视图(V) 搜索(S) 工具(T) 文档(D)

打开 保存 撤消 恢复 剪贴板

boot.asm

```
org 07c00h
mov ax, cs
mov ds, ax
mov es, ax
call DispStr
jmp $
DispStr:
mov ax, BootMessage
mov bp, ax
mov cx, 16
mov ax, 01301h
mov bx, 000ch
mov dl, 0
int 10h
mov ah, 0x0e
mov al, 0x0d
int 0x10
mov al, 0x0a
int 0x10
mov ax, 0x88
int 0x15
add ax, 1024
mov bx, 10
xor cx, cx
xor si, si
loopi: xor dx, dx
div bx
or dl, 0x30
mov [heap+si], dl
inc si
inc cx
cmp ax, 0
jnz loopi
dec si
mov ah, 0x0e
loopj: mov al, [heap+si]
int 0x10
dec si
loop loopj
mov ah, 0x0e
mov al, 0x0d
int 0x10
mov al, 0x0a
int 0x10
ret
BootMessage: db "Hello, OS world!"
heap: dw 10
times 510 - ($-$$) db 0
dw 0xaa55
```

纯文本 跳格宽度: 4 行 1, 列 1 插入

BIOS 中断 INT 0x10 有很多输出功能，各个功能的入口是通过 CPU 寄存器 AH 的值来决定的。而 AX = 88H 时，我们调用 INT 15H 中断，内存大小（超出 1M）的部分会返回在 AX 中（不超过 63M，以 K 为单位）。然后通过压入 heap 数组以十进制输出。

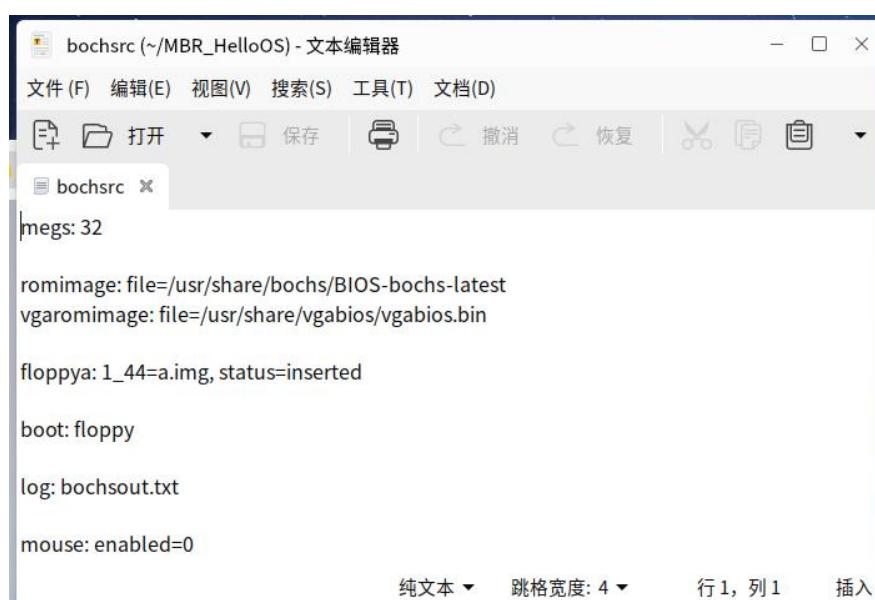
2) 生成二进制文件并写入虚拟软盘。

```
nasm boot.asm -o boot.bin
```

```
bximage 1 fd 1.44M a.img
```

```
dd if=boot.bin of=a.img bs=512 count=1 conv=notrunc
```

3) 写入 bochs 虚拟机配置文件 bochsrc



4) `bochs -f bochsrc` 启动虚拟机，在中断输入 c。查看继续执行，直到遇上断点前的输出。

3.2 编译启用新内核

开发环境：Ubuntu 22.04 LTS，内核版本 5.15.0-25-generic

1) 配置环境，安装相关依赖。

```
sudo apt update && sudo apt upgrade
```

```
sudo apt-get install libncurses5-dev libssl-dev
```

```
sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-
```

dev bison

```
sudo apt-get install build-essential openssl
```

```
sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex bison  
dwarves
```

```
sudo apt-get install libdw-dev
```

```
sudo apt-get install gcc gdb bison flex libncurses5-dev libssl-dev
```

```
sudo apt-get install gedit
```

```
sudo apt-get install zstd
```

2) 在 <https://www.kernel.org/> 下载稳定版内核。当前的最新稳定版为 2022-04-20 更新的 5.17.4 版本。

3) 解压缩并进入文件夹，生成配置文件。

```
sudo make mrproper
```

```
sudo make menuconfig
```

4) 修改.config 文件，删除含有 “debian” 的所有字符串。

5) 编译主体，编译模块，安装模块，安装内核，更新 GRUB。

```
sudo make bzImage -j4
```

```
sudo make modules -j4
```

```
sudo make INSTALL_MOD_STRIP=1 modules_install -j4
```

```
sudo mkinitramfs /lib/modules/5.17.4 -o /boot/initrd.img-5.17.4
```

```
sudo cp arch/x86/boot/bzImage /boot/vmlinuz-5.17.4
```

```
sudo cp System.map /boot/System.map-5.17.4
```

```
sudo update-grub2
```

6) 重启后查看系统内核版本。使用指令 `uname -r`。

3.3 Linux 增加系统调用

开发环境：Ubuntu 22.04 LTS，内核版本 5.15.0-25-generic

（本人在一开始多次尝试为银河麒麟 OS 更换内核，可以成功编译与生成，但更新 GRUB 后，配置文件未发生改变，新内核始终无法启用，上网查阅资料后发现有人成功为优麒麟操作系统更换内核，但尚未找到资料证明银河麒麟操作系统是否对内核更换做了某些限制——即无人发布自己成功更换内核及其步骤，也无人发布其不能更换内核。实际上通过查找资料发现，优麒麟与银河麒麟操作系统没有任何关系，UbuntuKylin 是 Ubuntu 社区中面向中文用户的 Ubuntu 衍生版本，中文名称优麒麟，难以证明其为国产操作系统。

但后来发现了问题所在，并成功在银河麒麟操作系统中实现了内核增加系统调用，编译并启用新内核。解决途径请参考五、实验错误排查与解决）

1) 配置环境，安装相关依赖。

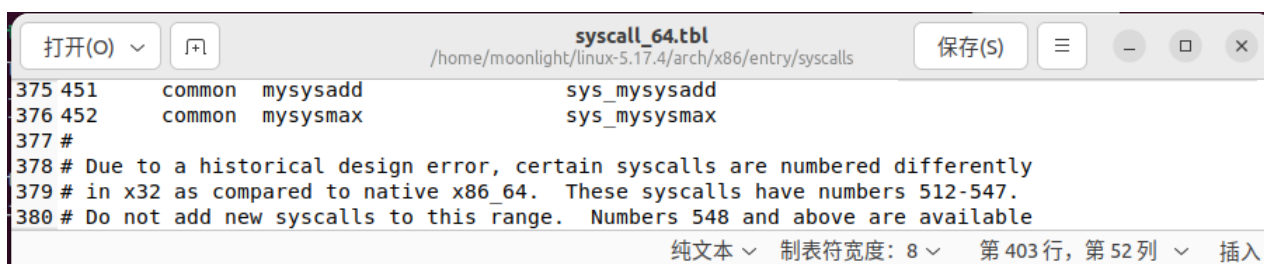
2) 在 <https://www.kernel.org/> 下载稳定版内核。当前的最新稳定版为 2022-04-20 更新的 5.17.4 版本。

3) 解压缩并进入文件夹，生成配置文件。

4) 修改 .config 文件，删除含有 “debian” 的所有字符串。

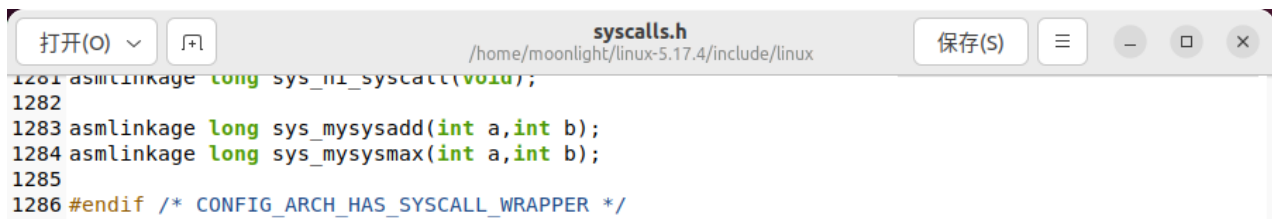
5) 增加系统调用

sudo gedit arch/x86/entry/syscalls/syscall_64.tbl 分配系统调用号。



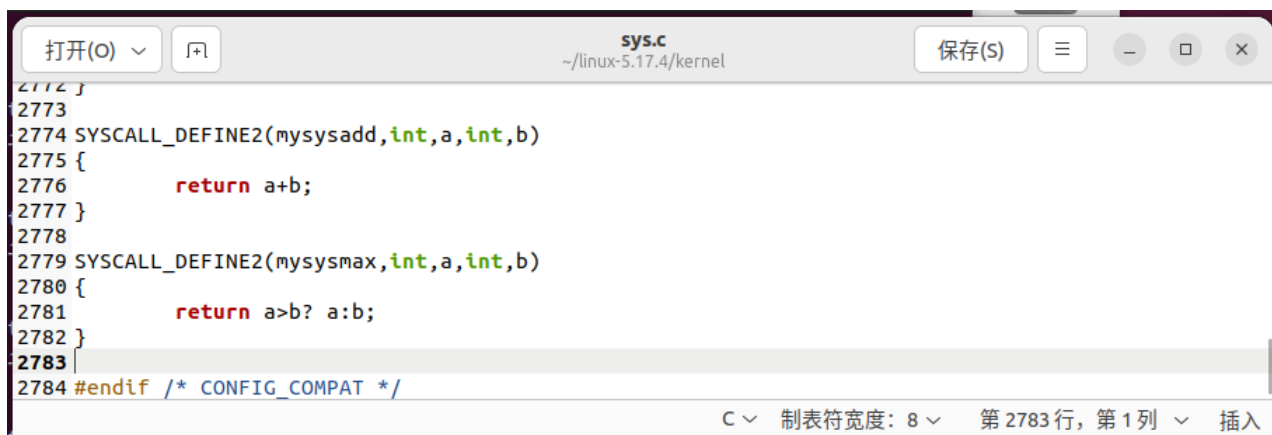
```
375 451      common  msysadd          sys_msysadd
376 452      common  msysmax          sys_msysmax
377 #
378 # Due to a historical design error, certain syscalls are numbered differently
379 # in x32 as compared to native x86_64. These syscalls have numbers 512-547.
380 # Do not add new syscalls to this range. Numbers 548 and above are available
```

sudo gedit include/linux/syscalls.h 声明系统调用。（注意位置）



```
1281 asmlinkage long sys_mysyscall(void);
1282
1283 asmlinkage long sys_mysysadd(int a,int b);
1284 asmlinkage long sys_mysysmax(int a,int b);
1285
1286 #endif /* CONFIG_ARCH_HAS_SYSCALL_WRAPPER */
```

gedit kernel/sys.c 实现系统调用函数

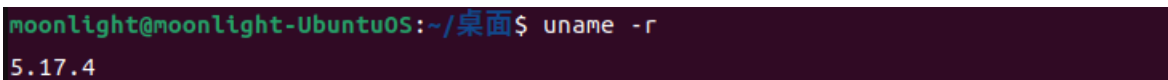


```
2772 }
2773
2774 SYSCALL_DEFINE2(mysysadd,int,a,int,b)
2775 {
2776     return a+b;
2777 }
2778
2779 SYSCALL_DEFINE2(mysysmax,int,a,int,b)
2780 {
2781     return a>b? a:b;
2782 }
2783
2784 #endif /* CONFIG_COMPAT */
```

保存并关闭。

6) 编译主体, 编译模块, 安装模块, 安装内核, 更新 GRUB。

7) 重启后查看系统内核版本, 并编写程序验证加入的系统调用。



```
moonlight@moonlight-UbuntuOS:~/桌面$ uname -r
5.17.4
```



```
1 #include <stdio.h>
2 #include <linux/kernel.h>
3 #include <sys/syscall.h>
4 #include <unistd.h>
5
6 int main () {
7     int nRet1 = syscall(451, 20,18); // nRet = 38
8     int nRet2 = syscall(452, 20,18, 4); // nRet = 20
9     printf("%d %d\n", nRet1, nRet2);
10    return 0;
11 }
```

代码简明清晰, 但是整体过程复杂。编译过程很长。

3.4 编写脚本，批量在文件中加入时间

开发环境：银河麒麟操作系统 v10 桌面版

编写脚本或批处理。在指定目录中的全部 txt 文件末尾追加或更新：用户名:日期和时间。



```
终端
文件(F) 编辑(E) 视图(V) 搜索(S) 终端(T) 帮助(H)
str_insert="#moonlight# $USER `date +%Y-%m-%d,%H:%M:%S`"
for ofile in $1/*.txt
do
    if [ `grep -c "#moonlight#" $ofile` -eq 0 ];then
        echo $str_insert >> $ofile
    else
        sed -i "/#moonlight#/c $str_insert" $ofile
    fi
done
cd $1
cat *.txt
~
~
~
~
1,1 全部
```

脚本定义了写入格式“#moonlight# username 2022-4-25,22:39”，通过特殊的字符串标识#moonlight#确定时需要追加还是更新。

for 语句遍历所有参数目录中的.txt 文件。若其中没有#moonlight#标识符，则利用重定向，按写入格式写入文件末尾。若存在标识符，则使用 sed 函数替换。

最后进入这个目录，将所有修改后的.txt 文件输出。

四、实验结果

4.1 编写 MBR 引导程序并测试



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
Hello, OS world! Bios (PCI) current-svn 07 Jan 2020
35464UGA/UBE Bios is released under the GNU LGPL

Please visit :
. http://bochs.sourceforge.net
. http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.11 BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +01
Options: apmbios pcibios pnpbios eltorito rombios

Press F12 for boot menu.
Booting from Floppy...

00000000000i[      ] lt_dlhandle is 0x563263634a70
00000000000i[PLUGIN] loaded plugin libbx_biosdev.so
00000000000i[      ] lt_dlhandle is 0x563263635410
00000000000i[PLUGIN] loaded plugin libbx_speaker.so
00000000000i[      ] lt_dlhandle is 0x56326363a350
00000000000i[PLUGIN] loaded plugin libbx_extfpuirq.so
00000000000i[      ] lt_dlhandle is 0x56326363ab20
00000000000i[PLUGIN] loaded plugin libbx_parallel.so
00000000000i[      ] lt_dlhandle is 0x56326363c780
00000000000i[PLUGIN] loaded plugin libbx_serial.so
00000000000i[      ] lt_dlhandle is 0x563263640b80
00000000000i[PLUGIN] loaded plugin libbx_gameport.so
00000000000i[      ] lt_dlhandle is 0x5632636413b0
00000000000i[PLUGIN] loaded plugin libbx_iodebug.so
00000000000i[      ] reading configuration from bochs.rc
00000000000i[      ] lt_dlhandle is 0x563263641e60
00000000000i[PLUGIN] loaded plugin libbx_x.so
00000000000i[      ] installing x module as the Bochs GL
00000000000i[      ] using log file bochsout.txt
Next at t=0
(0) [0x0000ffffffffff] f000:fff0 (unk. ctxt): jmpf 0xf000:
f0
<bochs:1> c
```

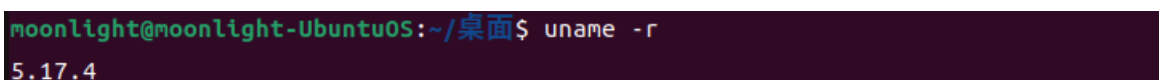
bochs -f bochs.rc 启动虚拟机，在中断输入 c。程序持续执行，直到遇上断点。期间使用 INT 10H 中断输出了字符串，也使用了 INT 15H 中断在 AX 中获得了内存容量，其加上 1024K 后，使用模拟的栈按照十进制输出，得到如上图的结果。

4.2 编译启用新内核

当我们打开 VMware 虚拟机时，长按 shift 键，进入 Ubuntu 高级选项，可以看到内核可供我们选择。



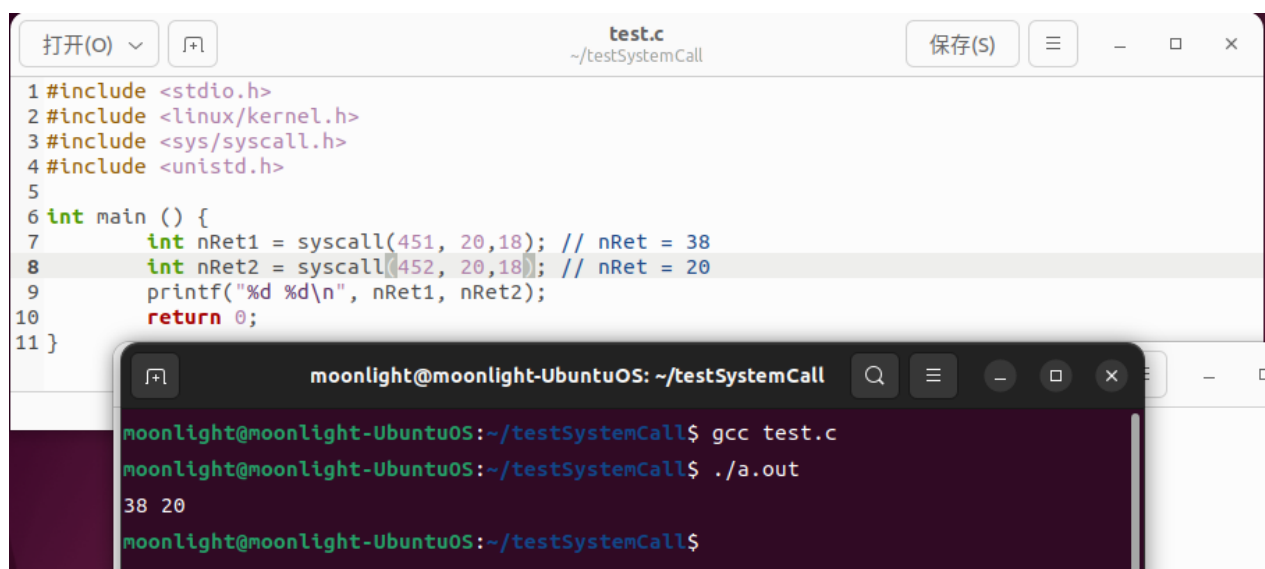
而且进入后使用 `uname -r` 指令观察，可证明我们已经成功编译启用了新内核。



```
moonlight@moonlight-UbuntuOS: ~/桌面$ uname -r
5.17.4
```

4.3 Linux 增加系统调用

我们测试我们编写的系统调用。



```
test.c
~/testSystemCall

1 #include <stdio.h>
2 #include <linux/kernel.h>
3 #include <sys/syscall.h>
4 #include <unistd.h>
5
6 int main () {
7     int nRet1 = syscall(451, 20, 18); // nRet = 38
8     int nRet2 = syscall(452, 20, 18); // nRet = 20
9     printf("%d %d\n", nRet1, nRet2);
10    return 0;
11 }
```

```
moonlight@moonlight-UbuntuOS: ~/testSystemCall
moonlight@moonlight-UbuntuOS:~/testSystemCall$ gcc test.c
moonlight@moonlight-UbuntuOS:~/testSystemCall$ ./a.out
38 20
moonlight@moonlight-UbuntuOS:~/testSystemCall$
```

因为我们的 451 号 syscall 即编写的 msysadd 系统调用函数，其返回两个参数的和。
452 号调用返回两个整型数的最大值。

4.3 编写脚本，批量在文件中加入时间

在运行脚本前，我们可以看到 text 文件夹内的内容如下。

```
moonlight@moonlight-Kylin:~/AddtimeTXT$ cd text
moonlight@moonlight-Kylin:~/AddtimeTXT/text$ ls
1.txt 2.txt 3.txt hello
moonlight@moonlight-Kylin:~/AddtimeTXT/text$ cat *txt
1
2
3
```

第一次运行 addtime.sh 后，结果如下。

```
moonlight@moonlight-Kylin:~/AddtimeTXT$ sh addtime.sh text
PlatformTheme Create "ukui"
ProxyStyle create "kysec_auth" "ukui"
Qt5UKUIStyle create "kysec_auth" "ukui-default"
1
#moonlight# moonlight 2022-04-25,22:55:55
2
#moonlight# moonlight 2022-04-25,22:55:55
3
#moonlight# moonlight 2022-04-25,22:55:55
```

在每个 txt 文件后加上了用户名与时间。

第二次运行后，我们可以看到只是更新了时间。

```
moonlight@moonlight-Kylin:~/AddtimeTXT$ sh addtime.sh text
1
#moonlight# moonlight 2022-04-25,22:57:29
2
#moonlight# moonlight 2022-04-25,22:57:29
3
#moonlight# moonlight 2022-04-25,22:57:29
```

五、实验错误排查和解决方法

4.1 编写 MBR 引导程序并测试

bochsrc 配置文件需要的字符格式比较严格，一开始报错，后通过上网查阅资料解决。

```
moonlight@moonlight-virtual-machine:~/桌面$ bochs -f bochsrc
=====
                Bochs x86 Emulator 2.6.11
                Built from SVN snapshot on January 5, 2020
                Timestamp: Sun Jan  5 08:36:00 CET 2020
=====
0000000000i[      ] LTDL_LIBRARY_PATH not set. using compile time default '/usr
/lib/bochs/plugins'
0000000000i[      ] BXSHARE not set. using compile time default '/usr/share/boc
hs'
0000000000i[      ] lt_dlhandle is 0x55cb3defdc40
0000000000i[PLUGIN] loaded plugin libbx_unmapped.so
0000000000i[      ] lt_dlhandle is 0x55cb3defea60
0000000000i[PLUGIN] loaded plugin libbx_biosdev.so
0000000000i[      ] lt_dlhandle is 0x55cb3deff400
0000000000i[PLUGIN] loaded plugin libbx_speaker.so
0000000000i[      ] lt_dlhandle is 0x55cb3df045a0
0000000000i[PLUGIN] loaded plugin libbx_extfpuirq.so
0000000000i[      ] lt_dlhandle is 0x55cb3df04d70
0000000000i[PLUGIN] loaded plugin libbx_parallel.so
0000000000i[      ] lt_dlhandle is 0x55cb3df069d0
0000000000i[PLUGIN] loaded plugin libbx_serial.so
0000000000i[      ] lt_dlhandle is 0x55cb3df0add0
0000000000i[PLUGIN] loaded plugin libbx_gameport.so
0000000000i[      ] lt_dlhandle is 0x55cb3df0b600
0000000000i[PLUGIN] loaded plugin libbx_iodebug.so
0000000000i[      ] reading configuration from bochsrc
0000000000p[      ] >>PANIC<< bochsrc:2: a bochsrc option needs at least one pa
rameter
0000000000e[SIM    ] notify called, but no bxevent_callback function is register
ed
0000000000e[SIM    ] notify called, but no bxevent_callback function is register
ed
=====
Bochs is exiting with the following message:
[      ] bochsrc:2: a bochsrc option needs at least one parameter
=====
0000000000i[SIM    ] quit_sim called with exit code 1
```

汇编语言是上学期的课程，但是却像是很久未见的老朋友，编写起汇编语言感觉很生疏。犯下了很多的错误。输出 Hello 字符串较为简单，但要输出一个数字就比较复杂，一开始出现了 bochs 虚拟机黑屏的现象，我仔细分析了代码，发现了其中的几个错误。

```
int 10h
mov al, 0x0d
int 0x10
mov ax, 0x88
int 0x15
mov al, 0x0a
int 0x10
add ax, 1024
```

此处代码本想输出换行后，进入 INT 15H 中断。但是很明显犯了错误。INT 10H 的中断参数没有赋，而且进入 INT 15H 中断的时机也不对。应该是如下的代码。

```
int 10h
mov ah, 0x0e
mov al, 0x0d
int 0x10
mov al, 0x0a
int 0x10
mov ax, 0x88
int 0x15
add ax, 1024
```

4.2 编译启用新内核

本实验之前做过一次作业，本以为会比较轻松。但是没想到仍然这么阴间。我多次尝试为银河麒麟 OS 更换内核，可以成功编译与生成，但更新 GRUB 后始终无法启用，上网查阅资料后发现有人成功为优麒麟操作系统更换内核，但尚未找到资料证明银河麒麟操作系统是否对内核更换做了某些限制——即无人发布自己成功更换内核及其步骤，也无人发布其不能更换内核。

```
BUILD arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready (#1)
```

```
moonlight@moonlight-virtual-machine:~/linux-5.17.4$ sudo make modules
[sudo] moonlight 的密码:
CALL scripts/checksyscalls.sh
CALL scripts/atomic/check-atomics.sh
DESCEND objtool
moonlight@moonlight-virtual-machine:~/linux-5.17.4$ sudo make bzImage -j4
DESCEND objtool
CALL scripts/atomic/check-atomics.sh
CALL scripts/checksyscalls.sh
CHK include/generated/compile.h
Kernel: arch/x86/boot/bzImage is ready (#1)
```

实际上通过查找资料发现，优麒麟与银河麒麟操作系统没有任何关系，UbuntuKylin 是

Ubuntu 社区中面向中文用户的 Ubuntu 衍生版本，中文名称优麒麟，难以证明其为国产操作系统。没有再去下载优麒麟操作系统，还是使用了 Ubuntu 系统。但我又发现更换过一次内核的 Ubuntu 再次编译生成新内核时，再安装模块时总是出错，通过卸载后重新安装 Ubuntu20.04LTS 解决了一切问题。

再后来，我的同学成功为银河麒麟操作系统更换了内核（没有增加系统调用），我对此感到十分诧异，在他的 history 指导下，仍然不能复现。我最终发现了问题在于银河麒麟操作系统的安全管理会要求用户在 30s 内点击，以允许一些危险的操作，而我可能忽视了一部分。我通过搜索关闭了银河麒麟操作系统的安全管理机制，按照步骤实现了内核的编译。



4.3 Linux 增加系统调用

如 4.2 上描述我成功为银河麒麟操作系统增加了系统调用处理函数。

```
moonlight@moonlight-VMware:~/testSyscall$ gcc test.c -o test.out
moonlight@moonlight-VMware:~/testSyscall$ ./test.out
38 48
```

4.4 编写脚本，批量在文件中加入时间

我对脚本和批处理程序的编写毫无经验，遇到此类问题只有大概的算法思路，不知如何 shell 脚本实现，通过借鉴网络上相似问题的代码，学习了 shell 脚本里的循环语句，判断语句以及一些特殊语句的写法。这些实验真的很有难度。

六、实验参考资料和网址

- (1) 教学课件：感谢华中科技大学软件学院苏曙光老师。
- (2) <http://www.zyiz.net/tech/detail-154923.html>
- (3) https://blog.csdn.net/weixin_43641850/article/details/104906726
- (4) https://blog.csdn.net/qq_46106285/article/details/121507087
- (5) <https://blog.csdn.net/zhangjs0322/article/details/10134299>
- (6) https://blog.csdn.net/qq_39819990/article/details/106605430
- (7) https://blog.csdn.net/qq_46106285/article/details/121507087
- (8) <https://eco.kylinos.cn/document/question/36.html>