《操作系统原理》实验报告

姓名	汪闻韵	学号	U202012056	专业班级	网安 2002	时间	2022.11.21

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（推荐银河麒麟或优麒麟，建议）。

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）下裁剪和编译 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。脚本参数 1 个：指定目录。脚本的作用是把指定目录中的全部文件的文件名加后缀，后缀是执行脚本时的日期和时分。例如：文件名“test”变成“test-2022-11-21-20-42”。

三、实验过程

3.1 银河麒麟系统编译启用 Linux 内核

下载优麒麟的镜像文件：

1. 官网下载：https://www.ubuntukylin.com/downloads/ （下载速度慢）
2. 镜像网站：官网底部（推荐），如图 3-1 所示。

图 3-1 优麒麟镜像网站

创建虚拟机硬盘空间至少大于 50G，否则后续编译内核会因存储空间不够而失败。硬件配置如图 3-2 所示。

图 3-2 虚拟机硬件配置

登录优麒麟后，注意一定要点击桌面的“安装“，否则试用版无法正常进行以下操作，系统重启会进行清空。

查看正在使用的内核版本，后面下载的内核版本最好不要高于目前版本，如图 3-3 所示。

```
#查看版本
uname -r
```

```
willow@willow-VMware-Virtual-Platform:~$ uname -r
5.10.0-5-generic
willow@willow-VMware-Virtual-Platform:~$
```

图 3-3 查看目前内核版本

安装编译的必要工具 gcc、gdb、bison、flex、libncurses5-dev、libssl-dev、libidn11 以及虚拟机的必备工具，如图 3-4 所示。

```
#安装编译所需工具
sudo apt install gcc gdb bison flex libncurses5-dev libssl-dev li
bidn11 build-essential
```

```
willow@willow-VMware-Virtual-Platform:~$ sudo apt install gcc gdb bison flex libncurses5-dev libssl-dev libidn11 build-e
ssential
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
gcc 已经是最新版 (4:9.3.0-11.185.1kylin2k6)。
gdb 已经是最新版 (9.1-0kylin1)。
libidn11 已经是最新版 (1.33-2.2kylin2)。
libidn11 已设置为手动安装。
下列软件包是自动安装的并且现在不需要了：
  libyaml-cpp0.6 localechooser-data user-setup
使用'sudo apt autoremove'来卸载它(它们)。
```

图 3-4 安装必要工具

从网上下载不高于目前内核版本的 kernel 源代码。

1.官网：https://www.kernel.org/ （下载速度慢）

2.镜像网站：http://ftp.sjtu.edu.cn/sites/ftp.kernel.org/pub/linux/kernel/（推荐）

此处下载 linux-5.4.3.tar.xz（可以自行选择其他版本），之后移动压缩包到 usr/src 目录下，解压后删除。

```
#移动压缩包到目标文件夹
sudo mv linux-5.4.3.tar.xz /usr/src
#解压缩
sudo tar -xf linux-5.4.3.tar.xz
#删除压缩包
sudo rm linux-5.4.3.tar.xz
```

将目前内核的配置拷贝到新内核的源码目录下，作为新内核的配置文件。其中，需要进入配置文件.config，查看一下 .config 文件是否变化，有的版本此操作会把前面双引号内置空的内容恢复，此时需要手动删除引号中的内容，如图 3-5 所示。

```
#进入源码目录
cd /usr/src/linux-5.4.3
#拷贝
sudo cp -v /boot/config-$(uname -r) .config
#查看文件
sudo vi .config
:/CONFIG_SYSTEM_TRUSTED_KEYS
```

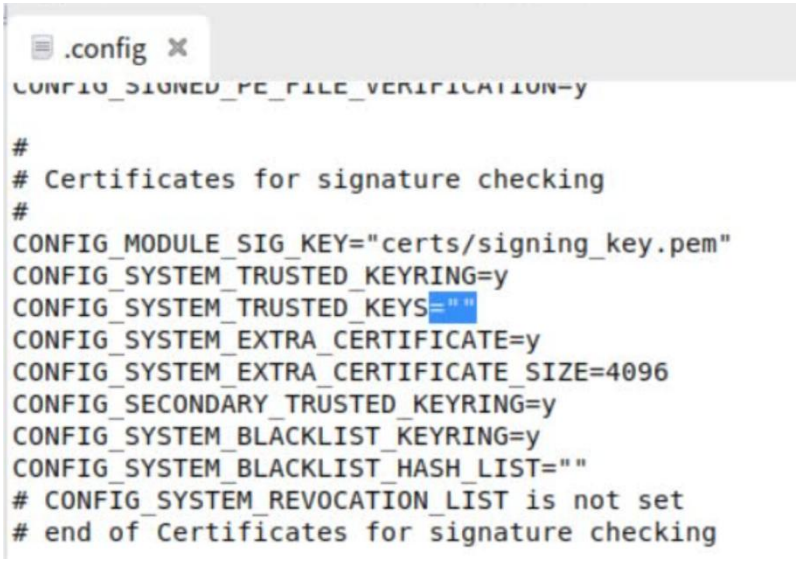

图 3-5 查看文件.config

使用 make menuconfig 个性化配置——实际实验过程中没有修改，直接使用默认配置即可。显示如下界面后，点击<Exit>即可，如图 3-6 所示。

```
sudo make menuconfig
```

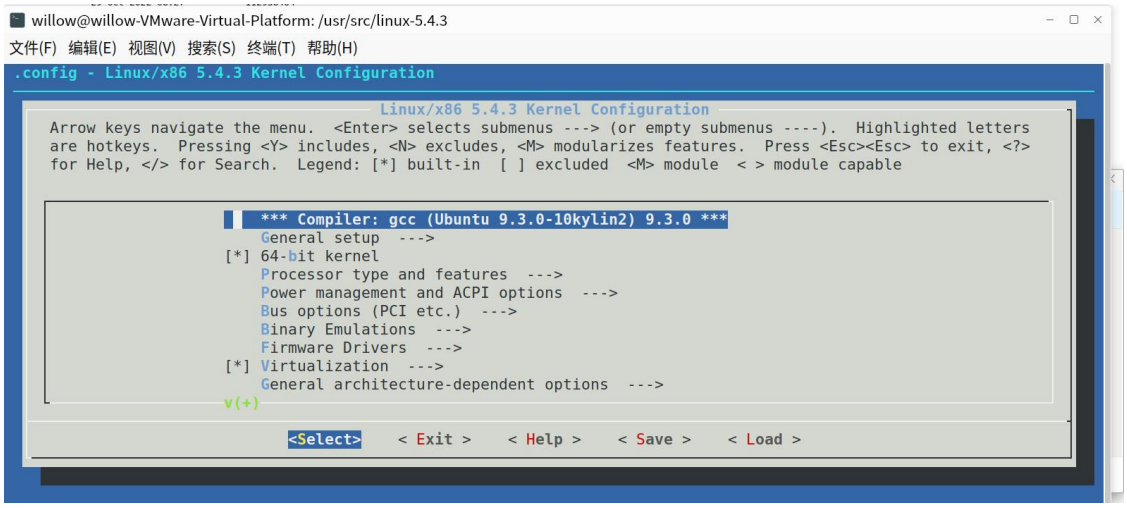

图 3-6 配置.config

进入源码目录，使用 make 命令编译内核，同时借助 -jn 加快编译速度，如图 3-7 所示。其中，n 是要生成的作业数，通常的做法是每个处理器产生一个或两个作业，可以查看虚拟机硬件配置（见图 3-2）中的处理器总分配内核数。此阶段比较耗时。

```
#4个作业同时进行编译
sudo make -j4
```

```
  LD [M]  sound/soc/xilinx/snd-soc-xlnx-i2s.ko
  LD [M]  sound/soc/xilinx/snd-soc-xlnx-spdif.ko
  LD [M]  sound/soc/xtensa/snd-soc-xtfpga-i2s.ko
  LD [M]  sound/soc/zte/zx-tdm.ko
  LD [M]  sound/soundcore.ko
  LD [M]  sound/synth/emux/snd-emux-synth.ko
  LD [M]  sound/synth/snd-util-mem.ko
  LD [M]  sound/usb/6fire/snd-usb-6fire.ko
  LD [M]  sound/usb/bcd2000/snd-bcd2000.ko
  LD [M]  sound/usb/caiaq/snd-usb-caiaq.ko
  LD [M]  sound/usb/hiface/snd-usb-hiface.ko
  LD [M]  sound/usb/line6/snd-usb-line6.ko
  LD [M]  sound/usb/line6/snd-usb-pod.ko
  LD [M]  sound/usb/line6/snd-usb-podhd.ko
  LD [M]  sound/usb/line6/snd-usb-toneport.ko
  LD [M]  sound/usb/line6/snd-usb-variax.ko
  LD [M]  sound/usb/misc/snd-ua101.ko
  LD [M]  sound/usb/snd-usb-audio.ko
  LD [M]  sound/usb/snd-usbmidi-lib.ko
  LD [M]  sound/usb/usx2y/snd-usb-us122l.ko
  LD [M]  sound/usb/usx2y/snd-usb-usx2y.ko
  LD [M]  sound/x86/snd-hdmi-lpe-audio.ko
  LD [M]  sound/xen/snd_xen_front.ko
willow@willow-VMware-Virtual-Platform:/usr/src/linux-5.4.3$ make mo
```

图 3-7 编译内核

安装内核模块，进一步安装内核模块，如图 3-8 所示。

```
sudo make modules_install
```

```
willow@willow-VMware-Virtual-Platform: /usr/src/linux-5.4.3
文件(F)  编辑(E)  视图(V)  搜索(S)  终端(T)  帮助(H)
  INSTALL sound/soc/xilinx/snd-soc-xlnx-spdif.ko
  INSTALL sound/soc/xtensa/snd-soc-xtfpga-i2s.ko
  INSTALL sound/soc/zte/zx-tdm.ko
  INSTALL sound/soundcore.ko
  INSTALL sound/synth/emux/snd-emux-synth.ko
  INSTALL sound/synth/snd-util-mem.ko
  INSTALL sound/usb/6fire/snd-usb-6fire.ko
  INSTALL sound/usb/bcd2000/snd-bcd2000.ko
  INSTALL sound/usb/caiaq/snd-usb-caiaq.ko
  INSTALL sound/usb/hiface/snd-usb-hiface.ko
  INSTALL sound/usb/line6/snd-usb-line6.ko
  INSTALL sound/usb/line6/snd-usb-pod.ko
  INSTALL sound/usb/line6/snd-usb-podhd.ko
  INSTALL sound/usb/line6/snd-usb-toneport.ko
  INSTALL sound/usb/line6/snd-usb-variax.ko
  INSTALL sound/usb/misc/snd-ua101.ko
  INSTALL sound/usb/snd-usb-audio.ko
  INSTALL sound/usb/snd-usbmidi-lib.ko
  INSTALL sound/usb/usx2y/snd-usb-us122l.ko
  INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
  INSTALL sound/x86/snd-hdmi-lpe-audio.ko
  INSTALL sound/xen/snd_xen_front.ko
  DEPMOD  5.4.3
willow@willow-VMware-Virtual-Platform:/usr/src/linux-5.4.3$
```

图 3-8 安装模块

模块安装完成后，进行内核安装，如图 3-9 所示。

```
sudo make install
```

```
I: Set the RESUME variable to override this.
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.4.3 /boot/vmlinuz-5
.4.3
waiting update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
正在生成 grub 配置文件 ...
找到主题: /usr/share/grub/themes/UKUI/theme.txt
找到 Linux 镜像: /boot/vmlinuz-5.10.0-5-generic
找到 initrd 镜像: /boot/initrd.img-5.10.0-5-generic
找到 Linux 镜像: /boot/vmlinuz-5.4.3
找到 initrd 镜像: /boot/initrd.img-5.4.3
找到 Linux 镜像: /boot/vmlinuz-5.4.3.old
找到 initrd 镜像: /boot/initrd.img-5.4.3
找到 initrd 镜像: /boot/initrd.img-5.10.0-5-generic
完成
willow@willow-VMware-Virtual-Platform:/usr/src/linux-5.4.3$
```

图 3-9 安装内核

将新安装的内核设置为引导，并更新 grub 引导程序，如图 3-10 所示。

```
#结尾为自己内核的版本
sudo update-initramfs -c -k 5.4.3
#更新 grub
sudo update-grub
```

```
willow@willow-VMware-Virtual-Platform:/usr/src/linux-5.4.3$ sudo update-grub
waiting update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
正在生成 grub 配置文件 ...
找到主题: /usr/share/grub/themes/UKUI/theme.txt
找到 Linux 镜像: /boot/vmlinuz-5.10.0-5-generic
找到 initrd 镜像: /boot/initrd.img-5.10.0-5-generic
找到 Linux 镜像: /boot/vmlinuz-5.4.3
找到 initrd 镜像: /boot/initrd.img-5.4.3
找到 Linux 镜像: /boot/vmlinuz-5.4.3.old
找到 initrd 镜像: /boot/initrd.img-5.4.3
找到 initrd 镜像: /boot/initrd.img-5.10.0-5-generic
完成
```

图 3-10 更新 grub

重启虚拟机，选择 高级选项——对应新内核的版本 进行启动，此处为 5.4.3，如图 3-11 所示。

图 3-11 启动新内核

3.2 添加系统调用并启用新内核

需要修改的文件如下：

1.系统调用：linux-5.4.3/kernel/sys.c

2.系统调用函数声明：linux-5.4.3/include/linux/syscalls.h

3. ID：linux-5.4.3/arch/x86/entry/syscalls/syscall_64.tbl

4. ID 声明：linux-5.4.3/include/uapi/asm-generic/unistd.h

需要修改的代码如下：

1.系统调用：linux-5.4.3/kernel/sys.c

```
SYSCALL_DEFINE2(SSD_Add,int,x,int,y){
        return x+y;
}

SYSCALL_DEFINE3(SSD_Max,int,a,int,b,int,c){
        if(a>b) b=a;
        if(b>c) c=b;
        return c;
}
```

2.系统调用函数声明：linux-5.4.3/include/linux/syscalls.h

```
asmlinkage long sys_SSD_Add(int x, int y);
asmlinkage long sys_SSD_Max(int a, int b, int c);
```

3.ID：linux-5.4.3/arch/x86/entry/syscalls/syscall_64.tbl

```
548     64      SSD_Add                 __x64_sys_SSD_Add
549     64      SSD_Max                 __x64_sys_SSD_Max
```

4. ID 声明：linux-5.4.3/include/uapi/asm-generic/unistd.h

```
#define __NR_ssd_add 548
__SYSCALL(__NR_ssd_add, sys_SSD_Add)
#define __NR_ssd_max 549
__SYSCALL(__NR_ssd_max, sys_SSD_Add)
```

如下图 3-12 所示。

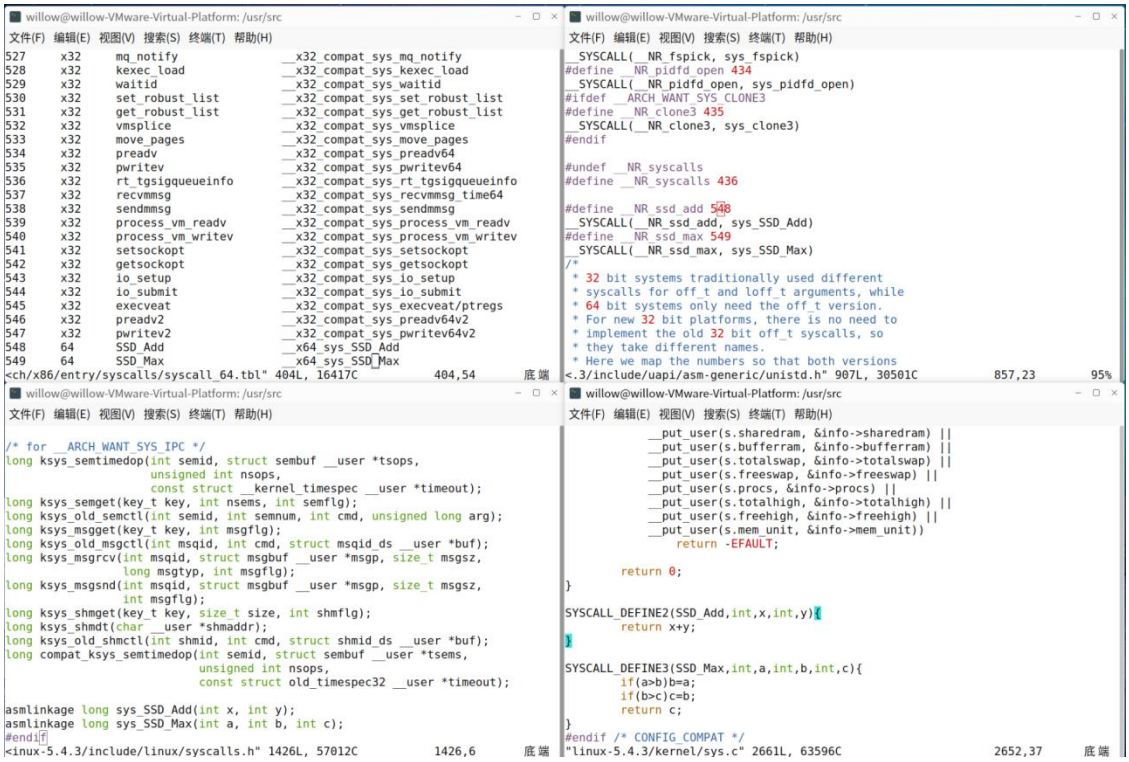

图 3-12 增加系统调用

重新编译，与实验二基本相同，编译前清理一下上一次残留的编译文件即可。

```
sudo make mrproper
sudo make menuconfig
sudo make -j4
sudo make modules_install
sudo make install
sudo update-initramfs -c -k 5.4.3
sudo update-grub
reboot
```

调用新增函数进行测试

测试文件：test.c

```
#include <unistd.h>
#include <sys/syscall.h>
#include <stdio.h>
int main(int argc, char *argv[])
{
    long ret;
    ret = syscall(549,1,2,3);   //Max
    printf("ret:%ld\n",ret);
    ret = syscall(549,7,6,5);   //Max
    printf("ret:%ld\n",ret);
    ret = syscall(548,4,6);     //Add
    printf("ret:%ld\n",ret);
    ret = syscall(549,7,9,8);   //Max
    printf("ret:%ld\n",ret);
}
```

3.3 编写批处理程序

建立文件，test1.txt 与 test2.txt，并初始化文本文件的内容，如图 3-13 所示。

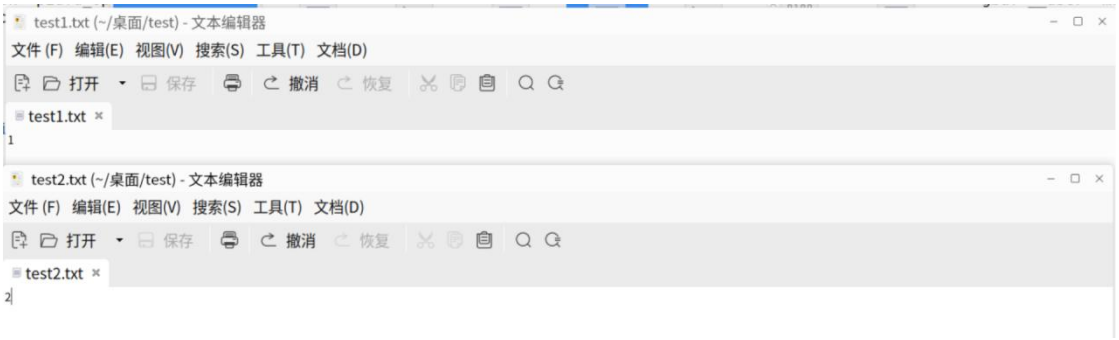

图 3-13 初始化文件

脚本文件“myshell2.sh”源代码如下，将待测试的 txt 文件与脚本文件放置于同一目录下，进行测试，如图 3-14 所示。

```
#设置特殊标识设为#OK#，表示已写入
str_insert="#OK# $USER `date +%Y-%m-%d,%H:%M:%S`"

#用$1接收第一个命令行参数(目录)；*.txt 检查文件后缀是否为txt
for ofile in $1/*.txt
do
    if [ `grep -c "#OK#" $ofile` -eq 0 ];then
        #未写入，向文件中追加写入字符串 str_insert
        echo $str_insert >> $ofile
    else
        #已写入，用 sed 更新字符串
        sed -i "/#OK#/c $str_insert" $ofile
    fi
done
```

```
str_insert="#OK# $USER `date +%Y-%m-%d,%H:%M:%S`"
for ofile in $1/*.txt
do
        if [ `grep -c "#OK#" $ofile` -eq 0 ];then
                echo $str_insert >> $ofile
        else
                sed -i "/#OK#/c $str_insert" $ofile
        fi
done
```

图 3-14 脚本文件

四、实验结果

4.1 银河麒麟系统编译启用 Linux 内核

启动完毕后，查看目前内核版本，若为新内核版本，则启动成功。如图 4-1 所示。

```
uname -r
```

```
willow@willow-VMware-Virtual-Platform:~/桌面$ uname -r
5.4.3
willow@willow-VMware-Virtual-Platform:~/桌面$
```

图 4-1 启用新内核

4.2 添加系统调用并启用新内核

编译运行编写的 test.c 文件进行测试。

```
sudo gcc -o test test.c
sudo ./test
```

输出如下图 4-2 所示。

图 4-2 新增系统调用的测试结果

4.3 编写批处理程序

运行脚本文件，其中，第一参数表示选择在该目录下的 txt 文件。

```
#第一参数"."表示选择该目录下的txt 文件
./myshell2.sh .
```

图 4-3 运行脚本文件

在目标目录下的所有 txt 文件新增一行 “#OK# willow 2022-11-28,19:19:48”。如图 4-4 所示。

图 4-4 第一次运行结果

再次运行，不再追加新行，而是刷新变为目前时间，“#OK# willow 2022-11-28,19:23:11”。如图 4-5 所示。

图 4-5 第二次运行结果

五、实验错误排查和解决方法

5.1 银河麒麟系统编译启用 Linux 内核

Q：重启时没有出现选择内核的界面（Unbuntu 可能会出现）

A：修改文件/etc/default/grub，添加如下两条命令，并注释掉 GRUB_TIMEOUT_STYLE，这一段代码，设置 GRUB_TIMEOUT 值为 30，意思是开机前留下 30s 的时间进入 grub 选择界面。随后更新 grub 并重启。

```
GRUB_SAVEDEFAULT=true
GRUB_DEFAULT=saved

# GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=30

sudo update-grub
sudo reboot
```

5.2 添加系统调用并启用新内核

Q：出现报错“ undefined reference to ‘xxx’ ”，如图 5-1 所示。

A：在 linux4.17 以后，添加系统调用必须以 __x64_sys_开头，将 syscall_64.tbl 中的系统调用改函数名开头即可。

```
  LD      vmlinux.o
  MODPOST vmlinux.o
  MODINFO modules.builtin.modinfo
  LD      .tmp_vmlinux1
ld: arch/x86/entry/syscall_64.o:(.rodata+0x2240): undefined reference to `sys_SSD_Add'
ld: arch/x86/entry/syscall_64.o:(.rodata+0x2248): undefined reference to `sys_SSD_Max'
make: *** [Makefile:1077: vmlinux] 错误 1
willow@willow-VMware-Virtual-Platform:/usr/src/linux-5.4.3$
```

图 5-1 第二次运行结果

Q：编译内核时无法编译完整的内核，很快就编译完成。

A：可能此时正在用新内核编译新内核。需要重启虚拟机，回到旧内核进行重新编译，才可以编译完整的新内核。

5.3 编写批处理程序

Q：再次运行脚本时无法更新，只能在末尾添加新时间，无法达到实验要求。

A：每次添加最后一行时，添加特殊标识，表示该文件已被写入过后缀，再写入时，新后缀需要覆盖以前的老后缀。

Q：运行时找不到 shell 文件或者显示权限不够，如图 5-2 所示。

```
willow@willow-VMware-Virtual-Platform:~/桌面/lab1/shell$ ./myshell2.sh .
bash: ./myshell2.sh: 权限不够
willow@willow-VMware-Virtual-Platform:~/桌面/lab1/shell$
```

图 5-2 权限不够终端显示

A：先新建了一个 txt 文件，之后再改的后缀名，故该脚本文件没有执行的权限，给脚本赋予权限即可。

六、实验参考资料和网址

（1）教学课件

（2）实验学习：https://blog.csdn.net/qq_46106285/article/details/121507087

（3）https://blog.csdn.net/weixin_46584887/article/details/125973263

（4）https://blog.csdn.net/qq_44765221/article/details/111087361