《操作系统原理》实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 邬雪菲 | 学号 | U202112131 | 专业班级 | 网安2104 | 时间 | 2023.11.21 |

**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-

2023-11-21-20-42”.

**三、实验过程**

**3.1 编写MBR引导程序**

1）环境配置

实验环境：VMware Workstation Pro 17，优麒麟20.04

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：在官网底部有多重选择（最后选择了华为云，速度较快）

创建虚拟机，内存设置为4G，硬盘空间尽量大，防止后续编译内核因存储空间不足而失败。（之后做任务二时编译了几个小时，怀疑是内存和硬盘开太小了，扩展了硬盘后做任务三，编译速度有所提高）



登录优麒麟后，点击桌面的“安装Ubuntu“，否则试用版无法正常进行后续操作，系统重启会进行清空（密码shiftw041）



安装完成后登录，打开终端，使用如下命令更新软件源，否则无法正常安装必要工具：

sudo apt-get update

sudo apt-get upgrade

2）安装NASM

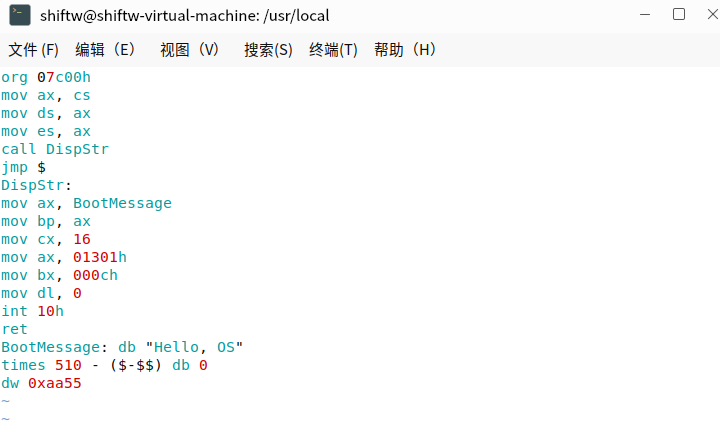
使用命令sudo apt install nasm

3）安装Bochs

使用命令sudo apt-get install vgabios bochs bochs-x bximage

4）编写引导扇区程序

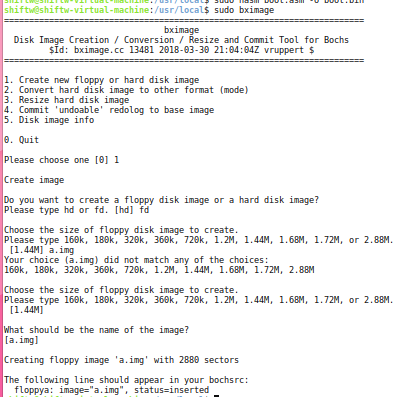
切换到usr/local/目录，创建sudo vim boot.asm，编写所提供的引导程序



编写完成后使用sudo nasm boot.asm -o boot.bin命令编译

5）创建虚拟软盘

通过如下sudo bximage命令创建虚拟软盘，选择生成类型为fd，设置软盘映像文件名，当前目录成功生成a.img软盘映像：



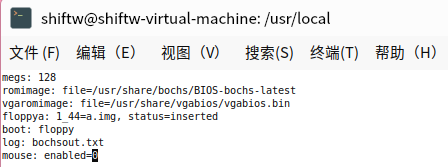
6）将引导扇区写入软盘

通过sudo dd if=boot.bin of=a.img bs=512 count=1 conv=notrunc命令将扇区写入软盘



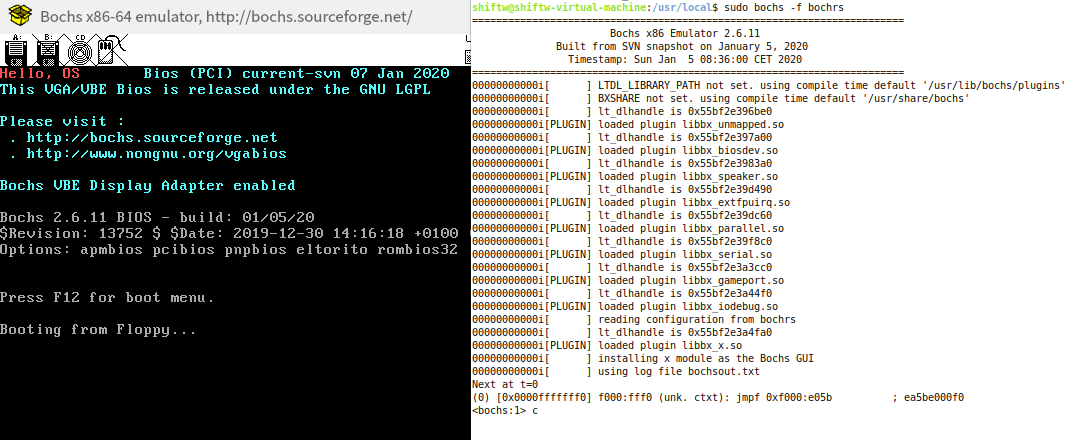
7）配置Bochs

创建bochrs文件，编写以下配置信息：



8）启动Bochs虚拟机

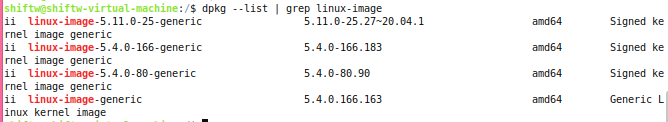
通过sudo bochs -f bochrs命令执行启动后返回终端，输入c按回车，可以看到执行成功，显示红色字体的Hello OS



**3.2 编译linux内核**

1）查看当前linux安装了哪些内核

dpkg --list | grep linux-image

****

uname -r查看正在使用的内核版本，后面下载的内核版本不能高于目前版本



2）下载linux内核源码并解压

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）

此处选择下载 linux-5.8.4.tar.xz ，之后移动压缩包到 usr/src目录下，解压后删除。

tar xf linux-5.8.4.tar.xz   
 sudo mv linux-5.8.4 /usr/src/  
 sudo rm linux-5.8.4.tar.xz

3）下载软件包

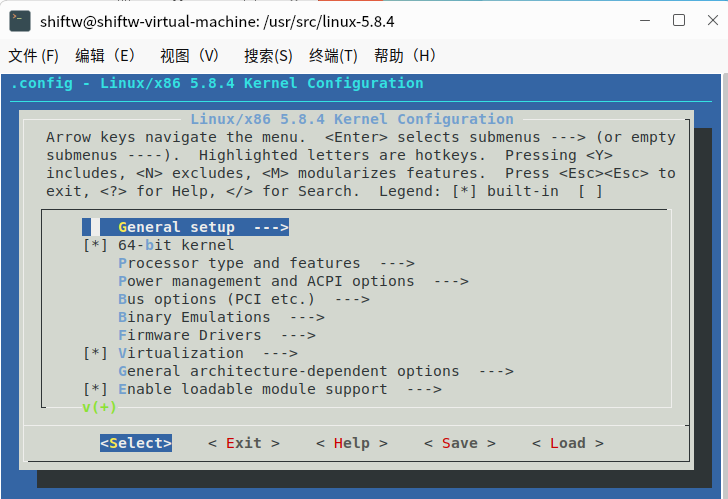
使用如下命令安装编译的必要工具gcc、gdb、bison、flex、libncurses5-dev、libssl-dev、libidn11以及虚拟机的必备工具：

sudo apt-get install gcc gdb bison flex libncurses5-dev libssl-dev libidn11 build-essential​

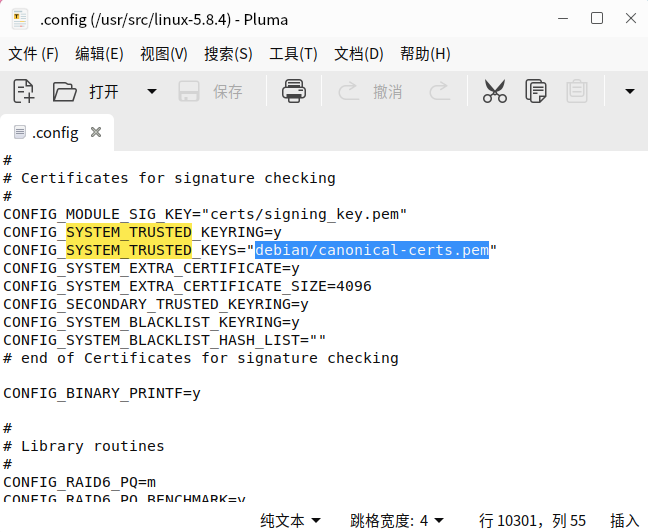
4）配置内核

sudo make mrproper #清除残留的.config和.o文件​  
 sudo make clean  
 sudo make menuconfig #打开配置内核的图形窗口使用个性化配置。

实际实验过程中无需修改，直接使用默认配置即可。显示如下界面后，点击<Exit>即可



修改配置文件sudo pluma .config，将CONFIG\_SYSTEM\_TRUSTED\_KEYS修改为空字符串



5）编译内核

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，采用sudo make -j4命令编译。n是要生成的线程数，通常每个处理器产生一个或两个线程较合适，否则编译进度过慢或者发生内存崩溃。可以查看虚拟机硬件配置中的处理器总分配内核数。由于虚拟机配置较低，此编译阶段非常耗时，可以挂着虚拟机去做别的事情。

6）安装内核模块

sudo make modules​  
 sudo make modules\_install​  
 sudo make install​

并将新安装的内核设置为引导，更新grub引导程序。

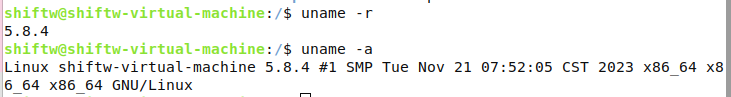
sudo update-initramfs -c -k 5.8.4  
 sudo update-grub2

7）进入新内核

终端输入reboot重启虚拟机，并在重启过程中按下tab+shift进入高级启动项，发现多出了新的内核版本，即为编译过后的新linux内核

选择新内核进入，终端查看目前内核版本发现是新安装的linux版本，且创建时间与编译开始时间相同，说明的确是新内核。



**3.3 系统调用添加**

1）添加系统调用号

通过如下命令进入系统调用号的配置

sudo vim /usr/src/linux-5.8.4/arch/x86/entry/syscalls/syscall\_64.tbl

在末尾添加两个新调用号

64 add sys\_add  
 64 max sys\_max



2）添加系统调用服务函数的声明

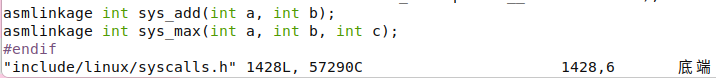
通过如下命令进入函数声明配置

sudo vim /usr/src/linux-5.8.4/include/linux/syscalls.h

添加两条新的函数声明

asmlinkage int sys\_add(int a, int b);

asmlinkage int sys\_max(int a, int b, int c);

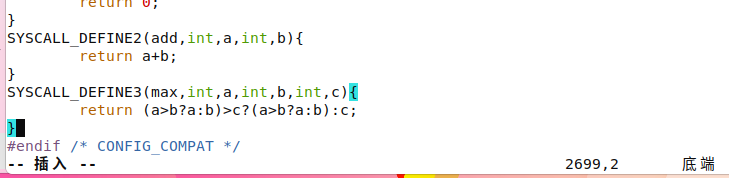


3）实现自己的系统调用服务函数

通过如下命令进入对函数的定义

sudo vim /usr/src/linux-5.8.4/kernel/sys.c

对两条新函数进行定义



1. 重新编译内核

参考任务二的步骤重新编译内核：

cd /usr/src/linux-5.8.4/  
 sudo make mrproper #清除残留的.config和.o文件​  
 sudo make clean  
 sudo make menuconfig #打开配置内核的图形窗口使用个性化配置。

实际实验过程中无需修改，直接使用默认配置点击<Exit>即可。

修改配置文件sudo pluma .config，将CONFIG\_SYSTEM\_TRUSTED\_KEYS修改为空字符串

sudo make -j8命令编译

sudo make modules​

sudo make modules\_install​

sudo make install​

sudo update-initramfs -c -k 5.8.4

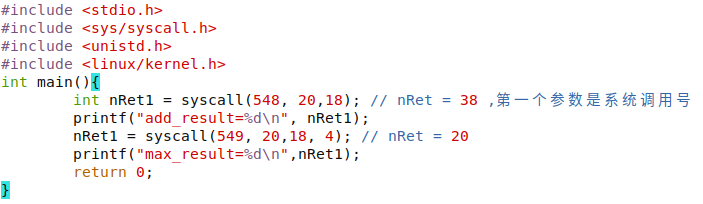
sudo update-grub2

终端输入reboot重启虚拟机，并按下tab+shift进入高级启动项，选择新内核进入



5）测试调用

编写c程序测试

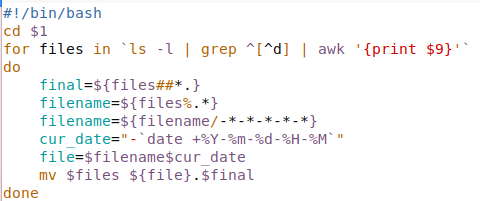


sudo vim test.c  
 gcc test.c -o test  
 ./test

**3.4 编写追加后缀脚本**

1）编写脚本

编写如下所示脚本文件作为批处理程序postfix.sh



2）测试程序

创建test文件夹存放测试文件若干进行测试，带参数执行批处理程序

sudo ./postfix.sh test/

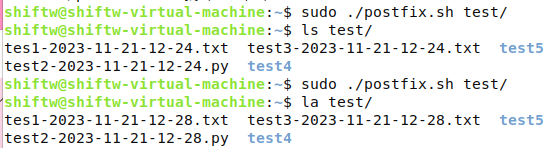
尝试运行批处理程序失败，报错显示没有此操作。

ls -l查看脚本文件权限



显示没有执行权限，故使用chmod u+x postfix.sh赋予权限

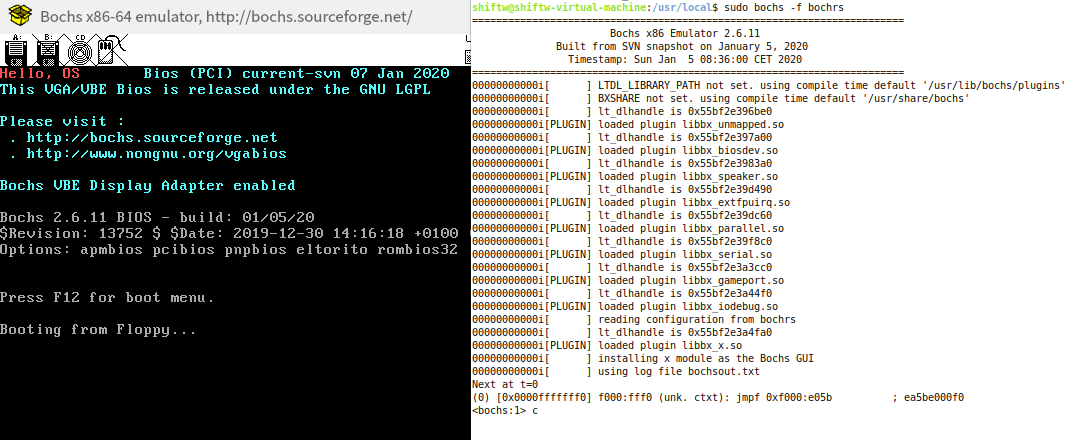
得到测试结果如下所示，可以看到test文件夹其中的两个子目录并没有被追加后缀，其余文件正常被追加后缀，追加时间格式为-年-月-日-时-分，重新运行批处理程序，后缀没有追加而是更新了时间。



**四、实验结果**

**4.1 编写MBR引导程序**

可以观察到执行成功，黑色启动界面中首行显示红色的Hello OS。

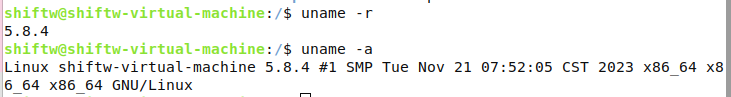


**4.2 编译linux内核**

重启虚拟机并进入高级选项，发现出现了新的内核版本，即为编译过后的新linux内核。

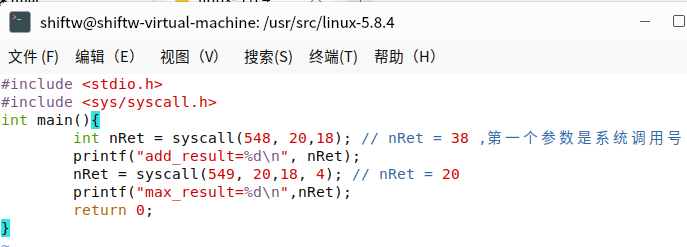
 

选择新内核进入，查看内核版本发现是新安装的linux版本，且创建时间也是新的，说明更换新内核成功。



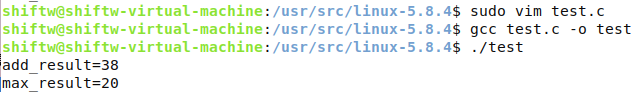
**4.3 添加系统调用**

编写c程序测试添加的系统调用函数，代码如下：



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

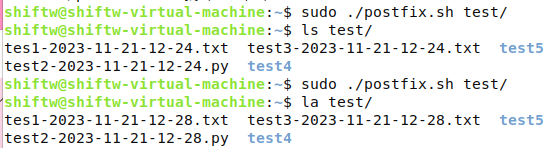
操作结果如下所示，能正确输出加和与最大值，可见调用成功！



**4.4 编写追加后缀脚本**

创建test文件夹存放测试文件若干进行测试，带参数执行批处理程序

得到测试结果如下所示，可以看到test文件夹其中的两个子目录并没有被追加后缀，其余文件正常被追加后缀，追加时间格式为-年-月-日-时-分，重新运行批处理程序，后缀没有追加而是更新了时间，说明批处理程序设计成功！



**五、实验错误排查和解决方法**

**5.1 编写MBR引导程序**

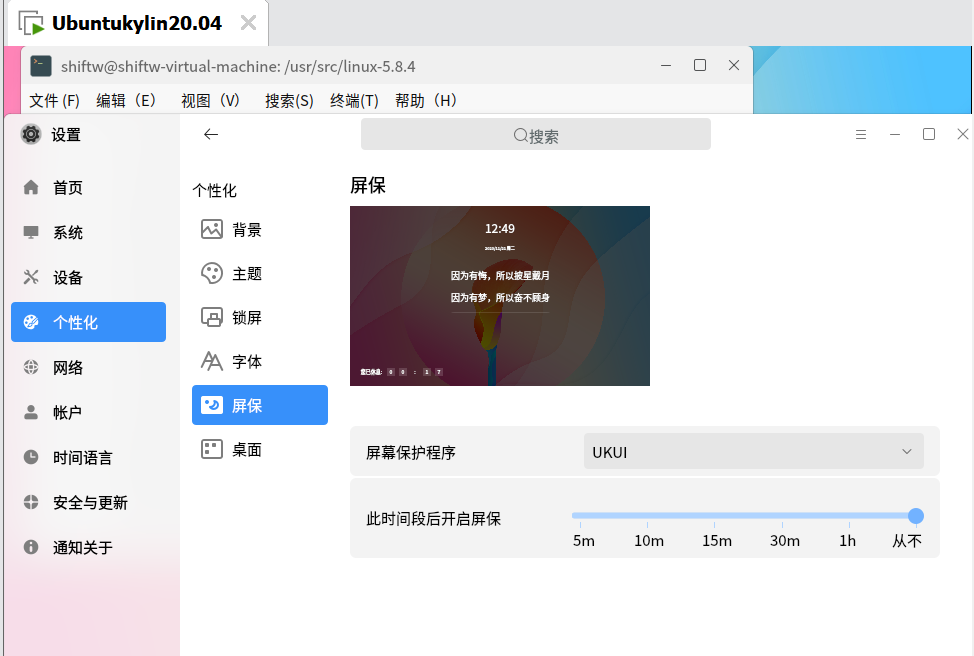
1. 进入引导程序后，没有出现预期的字样。因为还需要返回终端程序，执行按下c+enter的操作。

**5.2 编译linux内核**

1. 第一次尝试make编译的时候中间报错“没有规则可制作目标”，还需要更改源码文件夹里的.config配置文件，将CONFIG\_SYSTEM\_TRUSTED\_KEYS修改为空字符串，才能正常编译。且在查找该字段的时候vim的查找功能失效，不知道是什么原因，改用pluma查找成功。

2. 第一次用make -j4编译过程很慢，以及磁盘分配空间不够大。查到资料应该在一开始就尽量分配较大的磁盘和内存，同时真是磁盘空间也要预留足够，并且分配4个以上的内核。这是本次实验最大的疏忽，以为内存够就行了，之后再使用磁盘扩展似乎也用处不大了，导致编译阶段耗时巨大。下次创建虚拟机的时候还是要注意尽量配置更好的性能和大内存。

3. ubuntukylin虚拟机自动挂机锁屏，导致本就耗时巨大的编译过程频繁被打断。一开始尝试修改虚拟机电源相关选项，没有解决，搜索之后尝试关掉屏保的自动开启，成功解决问题。



1. 编译完成后用reboot命令重启，没有看到高级选项选择内核而是弹出登陆选项，先不要登录，而是点击右下角的电源键选择重启，按下tab+shift，即可看到高级选项。

**5.3 系统调用添加**

1. 添加系统调用函数源码时对SYSCALL\_DEFINEn( )中的n理解不清楚，n实际上指明了系统调用的参数个数，不可随意修改。

2. 一开始系统调用号的添加、系统调用函数的声明以及系统调用函数实现三者中的函数名没有对应好，导致添加失败报错，之后注意细节更改了就成功了。

**5.4 编写追加后缀脚本**

1. 一开始疑惑于如何区分文件夹和文件，想到的操作是遍历目标路径，对每个项目都进行判断是文件夹还是文件。后来发现可以使用只显示文件而不显示文件夹的命令ls -l | grep ^[^d] | awk '{print $9}'。

2. 对于如何区分加没加过后缀的文件，从参考资料中得到一个思路即打标记，这样只需要根据标记的有无决定是更新后缀还是加后缀，但增加了冗余的信息且没有完全满足要求。之后想到其实不必要打标记区分，对每个文件都执行删除后缀再添加后缀的操作就可以了。

3. 在处理文件后缀名的时候，考虑到文件名中可能有多个“.”干扰文件后缀的识别，将${files/.\*}改为${files%.\*}。

**六、实验参考资料和网址**

**（1）教学课件**

**（2）https://blog.csdn.net/qq\_46106285/article/details/121507087**

**（3）https://www.51cto.com/article/663841.html**

**（4）https://www.cnblogs.com/xiaocen/p/3717993.html**

**（5）https://www.cnblogs.com/zhang-jun-jie/p/9266858.html**

**（6）https://www.jb51.net/article/114157.htm**