

# 《操作系统原理》实验报告

姓名	侯竣	学号	U202116003	专业班级	密码 2101	时间	2023.12.1
----	----	----	------------	------	---------	----	-----------

## 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）下载剪和编译 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、实验环境和核心代码

所有环境均为优麒麟 20.04，内核版本是 5.11，编译工具 gcc11

### 3.1 用 NASM 编写引导程序，在 BOCHS 虚拟机中测试

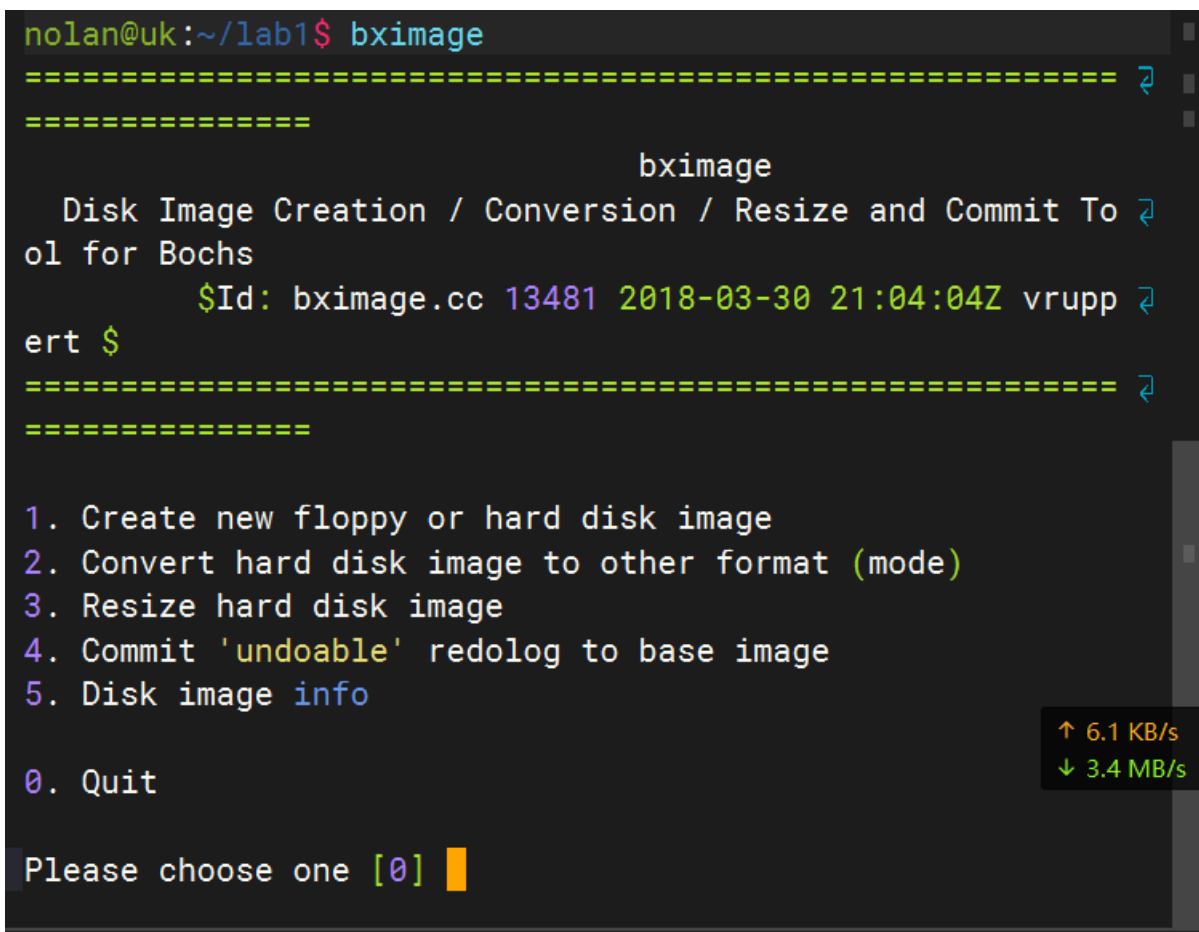
首先在虚拟机上安装 NASM, 和 Bochs 等工具:

执行 `sudo apt install nasm` 来安装 nasm

执行 `sudo apt-get install vgabios bochs bochs-x bximage` 安装 bochs

创建虚拟软盘:

输入 `bximage`, 如图:



```
nolan@uk:~/lab1$ bximage
=====
=====

                                bximage
  Disk Image Creation / Conversion / Resize and Commit To
  ol for Bochs
      $Id: bximage.cc 13481 2018-03-30 21:04:04Z vrupp
  ert $
=====
=====

1. Create new floppy or hard disk image
2. Convert hard disk image to other format (mode)
3. Resize hard disk image
4. Commit 'undoable' redolog to base image
5. Disk image info

0. Quit

Please choose one [0]
```

图 3-1

选择 1, 然后输入 `fd` 创建虚拟软盘, 然后一路回车即可

然后输入 `dd if=boot.bin of=a.img bs=512 count=1 conv=notrunc`, 将引导扇区写入软盘

然后写入 bochs 配置文件, `nano bochsrc` 创建配置文件

```
[20:17:05] 81 GNU nano 4.8 bochsrc
[20:17:05] 82 megs: 32
[20:17:05] 83
[20:17:05] 84 romimage: file=/usr/share/bochs/BIOS-bochs-latest
[20:17:05] 85 vgaromimage: file=/usr/share/vgabios/vgabios.bin
[20:17:05] 86
[20:17:05] 87 floppy: 1_44=a.img, status=inserted
[20:17:05] 88
[20:17:05] 89 boot: floppy
[20:17:05] 90
[20:17:05] 91 log: bochsout.txt
[20:17:05] 92
[20:17:05] 93 mouse: enabled=0
[20:17:05] 94
[20:17:05] 95
[20:17:05] 96
[20:17:05] 97
[20:17:05] 98
[20:17:05] 99
[20:17:05] 100
[20:17:05] 101
[20:17:05] 102
[20:17:05] 103
[20:17:05] 104
[20:17:05] 105
[20:17:05] 106 [ 已读取 12 行 ]
[20:17:05] 107 ^G 求助      ^O 写入      ^W 搜索      ^K 剪切文字
[20:17:05] 108 ^X 离开      ^R 读档      ^\ 替换      ^U 粘贴文字
```

↑ 0.1 KB/s  
↓ 2.3 KB/s

图 3-2

最后编写汇编代码, 创建 nano boot.asm

```
[20:19:25] 82  GNU nano 4.8      boot.asm
[20:19:25] 83  org 07c00h
[20:19:25] 84  mov ax, cs
[20:19:25] 85  mov ds, ax
[20:19:25] 86  mov es, ax
[20:19:25] 87  call DispStr
[20:19:25] 88  jmp $
[20:19:25] 89  DispStr:
[20:19:25] 90  mov ax, BootMessage
[20:19:25] 91  mov bp, ax
[20:19:25] 92  mov cx, 16
[20:19:25] 93  mov ax, 01301h
[20:19:25] 94  mov bx, 000ch
[20:19:25] 95  mov dl, 0
[20:19:25] 96  int 10h
[20:19:25] 97  ret
[20:19:25] 98  BootMessage: db "Hello, OS world!"
[20:19:25] 99  times 510 - ($-$$) db 0
[20:19:25] 100 dw 0xaa55
[20:19:25] 101
[20:19:25] 102
[20:19:25] 103
[20:19:25] 104
[20:19:25] 105
[20:19:25] 106
[20:19:25] 107
[20:19:25] 108
[20:19:25] 109
```

↑ 0.1 KB/s

↓ 0.0 KB/s

^G 求助

^O 写入

^W 搜索

^K 剪切文字

^X 离开

^R 读档

^\_ 替换

^U 粘贴文字

图 3-3

准备完毕后, 输入 `bochs -f bochsrc` 再输入 `c` 后, 启动虚拟机, 如图显示 Hello, OS world

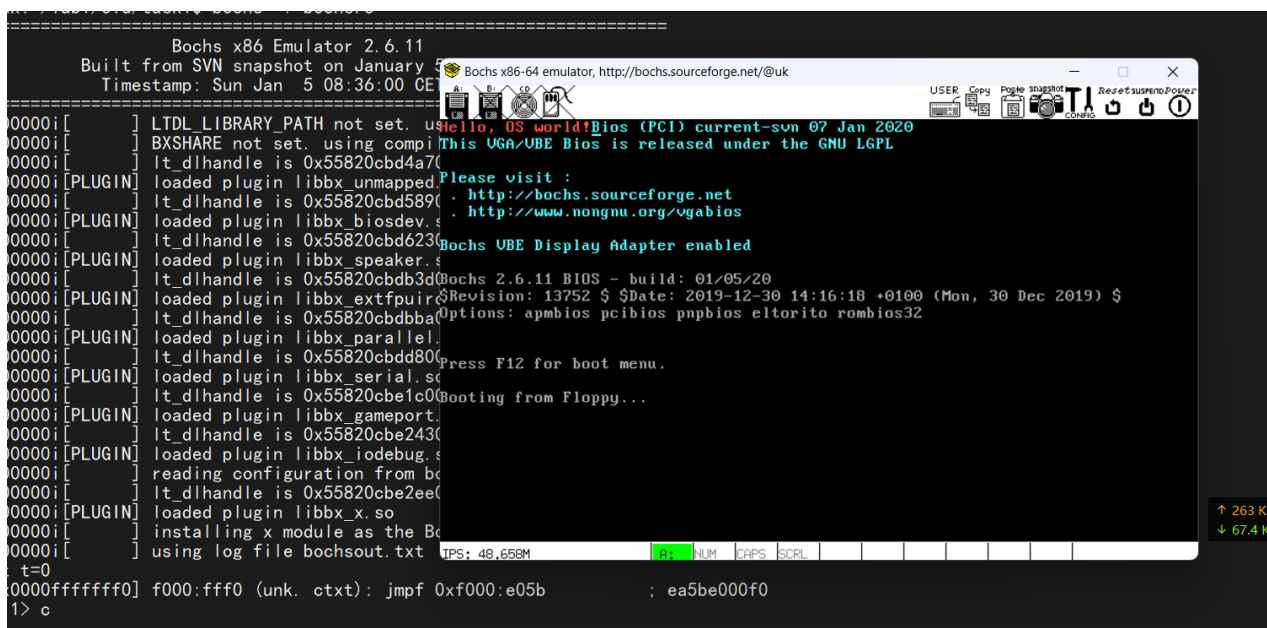


图 3-4

实验成功

### 3.2 在 Linux 下载剪和编译 Linux 内核并启用新内核

此实验主要是先下载内核，然后编译内核并启动，这里我选择了 5.15.139 版本内核

从官网下载内核到/usr/src，如图

```
nolan@uk:/usr/src$ ls
kylin-kmre-modules-dkms-1.0
linux-5.15.139
linux-headers-5.11.0-25-generic
linux-headers-5.4.0-166
linux-headers-5.4.0-166-generic
linux-hwe-5.11-headers-5.11.0-25
```

图 3-5

先安装必要的编译工具链:

```
apt-get install lib32ncurses5-dev gcc gdb bison flex libncurses5-dev libssl-dev libidn11-dev
libidn11 zlibc minizipbuild-essential openssl libelf-dev
```

然后 `make -j8`, 使用 8 线程编译

完成后分别执行:

`make modules`

`make modules_install`

`make install`

`update-grub2`

编译安装内核, 重启系统后, 输入 `uname -r` 查看内核版本:



```
nolan@uk:/usr/src$ uname -r
5.15.139
```

图 3-6

内核更新成功

### 3.3 增加系统调用, 启用新内核, 并测试。

首先增加系统调用表, 打开 `arch/x86/entry/syscalls/syscall_64.tbl` 加入两行系统调用:

```
[20:37:04] 194 GNU nano 4.8 syscall_64.tbl
[20:37:19] 195 529 x32 waitid compat_sys_waitid
[20:37:19] 196 530 x32 set_robust_list compat_sys_set_r>
[20:37:19] 197 531 x32 get_robust_list compat_sys_get_r>
[20:37:19] 198 532 x32 vmsplice sys_vmsplice
[20:37:19] 199 533 x32 move_pages sys_move_pages
[20:37:19] 200 534 x32 preadv compat_sys_pread>
[20:37:19] 201 535 x32 pwritev compat_sys_pwrit>
[20:37:19] 202 536 x32 rt_tgsigqueueinfo compat_sys_rt_tg>
[20:37:19] 203 537 x32 recvmmsg compat_sys_recvm>
[20:37:19] 204 538 x32 sendmmsg compat_sys_sendm>
[20:37:19] 205 539 x32 process_vm_readv sys_process_vm_r>
[20:37:19] 206 540 x32 process_vm_writev sys_process_vm_w>
[20:37:19] 207 541 x32 setsockopt sys_setsockopt
[20:37:19] 208 542 x32 getsockopt sys_getsockopt
[20:37:19] 209 543 x32 io_setup compat_sys_io_se>
[20:37:19] 210 544 x32 io_submit compat_sys_io_su>
[20:37:19] 211 545 x32 execveat compat_sys_execv>
[20:37:19] 212 546 x32 preadv2 compat_sys_pread>
[20:37:19] 213 547 x32 pwritev2 compat_sys_pwrit>
[20:37:19] 214 # This is the end of the legacy x32 range. Numbers 548 >
[20:37:19] 215 # not special and are not to be used for x32-specific sy>
[20:37:19] 216 548 64 add sys_add
[20:37:19] 217 549 64 max sys_max
[20:37:19] 218
[20:37:19] 219
[20:37:19] 220 ^G 求助 ^O 写入 ^W 搜索 ^K 剪切文字
[20:37:19] 221 ^X 离开 ^R 读档 ^\ 替换 ^U 粘贴文字
```

图 3-7

然后声明系统调用服务例程, 打开 include/linux/syscalls.h

```
[21:04:28] 943 GNU nano 4.8 syscalls.h
[21:04:34] 944 long compat_ksys_semtimedop(int semid, struct sembuf __u>
[21:04:34] 945 unsigned int nsops,
[21:04:34] 946 const struct old_timespec32 >
[21:04:34] 947 long __do_semtimedop(int semid, struct sembuf *tsems, un>
[21:04:34] 948 const struct timespec64 *timeout,
[21:04:34] 949 struct ipc_namespace *ns);
[21:04:34] 950
[21:04:34] 951 int __sys_getsockopt(int fd, int level, int optname, cha>
[21:04:34] 952 int __user *optlen);
[21:04:34] 953 int __sys_setsockopt(int fd, int level, int optname, cha>
[21:04:34] 954 int optlen);
[21:04:34] 955 asmlinkage int sys_add(int a,int b);
[21:04:34] 956 asmlinkage int sys_max(int a,int b,int c);
[21:04:34] 957 #endif
[21:04:34] 958
[21:04:34] 959
[21:04:34] 960
[21:04:34] 961
[21:04:34] 962
[21:04:34] 963
[21:04:34] 964
[21:04:34] 965
[21:04:34] 966
[21:04:34] 967
[21:04:34] 968
[21:04:34] 969 ^G 求助 ^O 写入 ^W 搜索 ^K 剪切文字
[21:04:34] 970 ^X 离开 ^R 读档 ^\ 替换 ^U 粘贴文字
```

图 3-8

然后在 kernel/sys.c 中实现自己的函数, 与步骤 2 相同, 编译安装修改后的内核

编写函数测试系统调用:



```
[21:07:31] 1153 GNU nano 4.8 task3.c
[21:07:31] 1154 // syscall_test.c
[21:07:31] 1155 #include <linux/kernel.h>
[21:07:31] 1156 #include <sys/syscall.h>
[21:07:31] 1157 #include <unistd.h>
[21:07:31] 1158 #include <stdio.h>
[21:07:31] 1159 int main(){
[21:07:31] 1160     int nRct1 = syscall(548,10,20);
[21:07:31] 1161     printf("syscall add(a,b) :%d\n",nRct1); // 30
[21:07:31] 1162     int nRct2 = syscall(549,100,102,88); //102
[21:07:31] 1163     printf("syscall max(a,b,c) :%d\n",nRct2);
[21:07:31] 1164 }
[21:07:31] 1165
[21:07:31] 1166
[21:07:31] 1167
[21:07:31] 1168
[21:07:31] 1169
[21:07:31] 1170
[21:07:31] 1171
[21:07:31] 1172
[21:07:31] 1173
[21:07:31] 1174
[21:07:31] 1175
[21:07:31] 1176
[21:07:31] 1177
[21:07:31] 1178 [ 已读取 11 行 ]
[21:07:31] 1179 ^G 求助      ^O 写入      ^W 搜索      ^K 剪切文字
[21:07:31] 1180 ^X 离开      ^R 读档      ^\ 替换      ^U 粘贴文字
                                     ↑ 0.3 KB/s
                                     ↓ 1.6 KB/s
```

图 3-9

```
nolan@uk:~/lab1/task3$ ./task3
syscall add(a,b) :30
syscall max(a,b,c) :102
```

图 3-9

结果正确

### 3.4 在 Linux 或 Windows 下，编写批处理重命名文件。

编写 shell 脚本，如图：

```

[21:23:01] 1195 nolan@uk:~/lab1/task4$ nano task4.sh
[21:23:38] 1196 nolan@uk:~/lab1/task4$ cat task4.sh
[21:23:38] 1197 #!/bin/bash
[21:23:38] 1198
[21:23:38] 1199 directory=$1
[21:23:38] 1200
[21:23:38] 1201 timestamp=$(date +"%Y-%m-%d-%H-%M")
[21:23:38] 1202
[21:23:38] 1203 for file in "$directory"/*; do
[21:23:38] 1204     if [ -f "$file" ]; then
[21:23:38] 1205         filename=$(basename "$file")
[21:23:38] 1206         extension="${filename##*."}"
[21:23:38] 1207         filename="${filename%.*}"
[21:23:38] 1208
[21:23:38] 1209         # 提取已有的时间戳
[21:23:38] 1210         existing_timestamp=$(echo "$filename" | grep -oE '[0-9]{4}-[0-9]{2}-[0-9]{2}-[0-9]{2}-[0-9]{2}$')
[21:23:38] 1211
[21:23:38] 1212         if [ -n "$existing_timestamp" ]; then
[21:23:38] 1213             # 文件名已有时间戳后缀，更新日期和时间
[21:23:38] 1214             new_filename="${filename%-$existing_timestamp}-${timestamp}.${extension}"
[21:23:38] 1215         else
[21:23:38] 1216             # 文件名没有时间戳后缀，添加完整的时间戳后缀
[21:23:38] 1217             new_filename="${filename}-${timestamp}.${extension}"
[21:23:38] 1218         fi
[21:23:38] 1219
[21:23:38] 1220         mv "$file" "$directory/$new_filename"

```

图 3-10

```

[21:24:16] 1227 nolan@uk:~/lab1/task4$ cd files
[21:24:17] 1228 nolan@uk:~/lab1/task4/files$ ls
[21:24:17] 1229 bytedence-2023-11-21-16-07.txt
[21:24:17] 1230 hust-2023-11-21-16-07.txt
[21:24:17] 1231 linux-2023-11-21-16-07.sys
[21:24:17] 1232 tencent-2023-11-21-16-07.txt
[21:24:17] 1233 testdir
[21:24:17] 1234 urk-2023-11-21-16-07.sys
[21:24:17] 1235 win10-2023-11-21-16-07.sys
[21:24:17] 1236 wuhan-2023-11-21-16-07.txt
[21:24:22] 1237 nolan@uk:~/lab1/task4/files$ cd ..
[21:24:22] 1238 nolan@uk:~/lab1/task4$ ls
[21:24:22] 1239 files files2 task4.sh
[21:24:27] 1240 nolan@uk:~/lab1/task4$ sh task4.sh files
[21:24:27] 1241 Renamed bytedence-2023-11-21-16-07 to bytedence-2023-12-04-21-24.txt
[21:24:27] -
[21:24:27] 1242 Renamed hust-2023-11-21-16-07 to hust-2023-12-04-21-24.txt
[21:24:27] -
[21:24:27] 1243 Renamed linux-2023-11-21-16-07 to linux-2023-12-04-21-24.sys
[21:24:27] -
[21:24:27] 1244 Renamed tencent-2023-11-21-16-07 to tencent-2023-12-04-21-24.txt
[21:24:27] -
[21:24:27] 1245 Renamed urk-2023-11-21-16-07 to urk-2023-12-04-21-24.sys
[21:24:27] 1246 Renamed win10-2023-11-21-16-07 to win10-2023-12-04-21-24.sys
[21:24:27] -
[21:24:27] 1247 Renamed wuhan-2023-11-21-16-07 to wuhan-2023-12-04-21-24.txt
[21:24:27] -
[21:24:27] 1248 nolan@uk:~/lab1/task4$

```

图 3-11

运行如图, 成功重命名文件

## 四、实验结果

### 4.1 用 NASM 编写引导程序, 在 BOCHS 虚拟机中测试



```
[21:07:31] 1153 GNU nano 4.8 task3.c
[21:07:31] 1154 // syscall_test.c
[21:07:31] 1155 #include <linux/kernel.h>
[21:07:31] 1156 #include <sys/syscall.h>
[21:07:31] 1157 #include <unistd.h>
[21:07:31] 1158 #include <stdio.h>
[21:07:31] 1159 int main(){
[21:07:31] 1160     int nRct1 = syscall(548,10,20);
[21:07:31] 1161     printf("syscall add(a,b) :%d\n",nRct1); // 30
[21:07:31] 1162     int nRct2 = syscall(549,100,102,88); //102
[21:07:31] 1163     printf("syscall max(a,b,c) :%d\n",nRct2);
[21:07:31] 1164 }
[21:07:31] 1165
[21:07:31] 1166
[21:07:31] 1167
[21:07:31] 1168
[21:07:31] 1169
[21:07:31] 1170
[21:07:31] 1171
[21:07:31] 1172
[21:07:31] 1173
[21:07:31] 1174
[21:07:31] 1175
[21:07:31] 1176
[21:07:31] 1177
[21:07:31] 1178 [ 已读取 11 行 ]
[21:07:31] 1179 ^G 求助      ^O 写入      ^W 搜索      ^K 剪切文字
[21:07:31] 1180 ^X 离开      ^R 读档      ^\ 替换      ^U 粘贴文字
                                     ↑ 0.3 KB/s
                                     ↓ 1.6 KB/s
```

图 4-3

```
nolan@uk:~/lab1/task3$ ./task3
syscall add(a,b) :30
syscall max(a,b,c) :102
```

图 4-4

编写测试脚本，利用系统调用计算 add 和 max 函数，结果正确

#### 4.4 在 Linux 或 Windows 下，编写批处理重命名文件。

```
[21:24:16] 1227 nolan@uk:~/lab1/task4$ cd files
[21:24:17] 1228 nolan@uk:~/lab1/task4/files$ ls
[21:24:17] 1229 bytedence-2023-11-21-16-07.txt
[21:24:17] 1230 hust-2023-11-21-16-07.txt
[21:24:17] 1231 linux-2023-11-21-16-07.sys
[21:24:17] 1232 tencent-2023-11-21-16-07.txt
[21:24:17] 1233 testdir
[21:24:17] 1234 urk-2023-11-21-16-07.sys
[21:24:17] 1235 win10-2023-11-21-16-07.sys
[21:24:17] 1236 wuhan-2023-11-21-16-07.txt
[21:24:22] 1237 nolan@uk:~/lab1/task4/files$ cd ..
[21:24:22] 1238 nolan@uk:~/lab1/task4$ ls
[21:24:22] 1239 files files2 task4.sh
[21:24:27] 1240 nolan@uk:~/lab1/task4$ sh task4.sh files
[21:24:27] 1241 Renamed bytedence-2023-11-21-16-07 to bytedence-2023-12-04-21-24.txt
[21:24:27] -
[21:24:27] 1242 Renamed hust-2023-11-21-16-07 to hust-2023-12-04-21-24.txt
[21:24:27] -
[21:24:27] 1243 Renamed linux-2023-11-21-16-07 to linux-2023-12-04-21-24.sys
[21:24:27] -
[21:24:27] 1244 Renamed tencent-2023-11-21-16-07 to tencent-2023-12-04-21-24.txt
[21:24:27] -
[21:24:27] 1245 Renamed urk-2023-11-21-16-07 to urk-2023-12-04-21-24.sys
[21:24:27] 1246 Renamed win10-2023-11-21-16-07 to win10-2023-12-04-21-24.sys
[21:24:27] -
[21:24:27] 1247 Renamed wuhan-2023-11-21-16-07 to wuhan-2023-12-04-21-24.txt
[21:24:27] -
[21:24:27] 1248 nolan@uk:~/lab1/task4$
```

图 4-5

如图运行脚本，重命名文件成功并且对于已经有时间戳的文件不会追加，而且没有命名子目录，实验成功

## 五、实验错误排查和解决方法

### 4.1 用 NASM 编写引导程序，在 BOCHS 虚拟机中测试

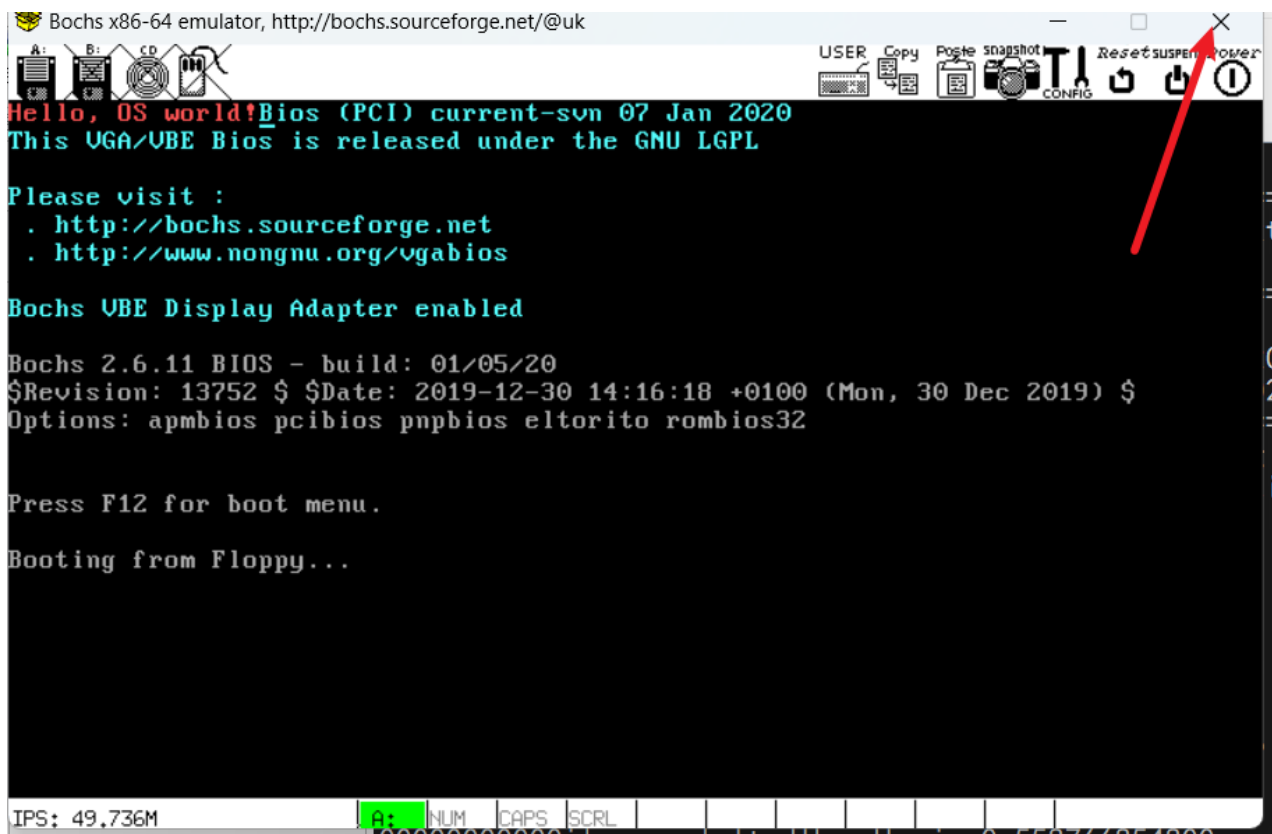


图 5-1

做实验的时候发现一个有趣的现象：退出虚拟机的时候如果按这个 X 会导致进程卡死，只能按 X 下面的关闭按钮才能成功关闭

不小心按了 X 就只能 kill 才能关闭 bochs 了

## 4.2 在 Linux 下载剪和编译 Linux 内核并启用新内核

## 4.3 增加系统调用，启用新内核，并测试。

编译内核的时候内核重启不成功，显示内核 panic，再次编译仍然出现这个问题，无果而终，然后通过对比其他成功的同学的 VMware，发现是我的 VMware 版本太老出现的问题，换成 VMware17 成功编译(当时忘记截图了，已经装上了 VMware17 就不好复现这个问题了，所以没有图)

## 4.4 在 Linux 或 Windows 下，编写批处理重命名文件。

一开始直接获取 shell 脚本所在的目录, 结果忘记过滤脚本自己, 给脚本命名了, 而固定目录不太优雅, 于是了解了 shell 的传参方法, 使用\$1 获取.sh 文件输入的第一个参数

运行时, 发现忘记过滤目录了, 把目录也重命名了一遍, 于是增加了目录判断命令

此外, 在调试时又发现, 对于没有后缀名的文件命名会重复一遍后缀名, 再次修了一遍 bug

## 六、实验参考资料和网址

### (1) 教学课件

(2) <https://www.cnblogs.com/leewithh/p/15434816.html> VMware 运行 Bochs 卡顿的一个解决办法

(3) <https://www.cnblogs.com/kuangke/p/14702349.html> Bochs 使用简单教程

(4) [https://blog.csdn.net/weixin\\_42250528/article/details/116631855](https://blog.csdn.net/weixin_42250528/article/details/116631855) vmware linux 编译内核,VMWare 编译 linux 内核的注意事项

(5) <https://blog.csdn.net/chenaqiao/article/details/17612675> (编译内核) Vmware 编译内核

(6) <https://blogs.vmware.com/affiliates/troubleshooting-kernel-panic> Troubleshooting Kernel Panic - VMware Blogs

(7) <https://www.runoob.com/linux/linux-tutorial.html> 菜鸟教程-linux 教程

(8) <https://www.runoob.com/w3cnote/shell-quick-start.html> 菜鸟教程-Shell 编程快速入门