



华中科技大学

操作系统原理课程设计报告

姓 名:

学 院:

专 业:

班 级:

学 号:

指导教师:

分数	
教师签名	

2020 年 2 月 21 日

目 录

1	实验一 熟悉和理解 Linux 编程环境	4
1.1	实验目的	4
1.2	实验内容	4
1.3	实验设计	4
1.3.1	开发环境.....	4
1.3.2	实验设计.....	4
1.4	实验调试.....	5
1.4.1	实验步骤.....	5
1.4.2	实验调试及心得.....	8
2	实验二 添加系统调用.....	9
2.1	实验目的	9
2.2	实验内容	9
2.3	实验设计	9
2.3.1	开发环境.....	9
2.3.2	实验设计.....	9
2.4	实验调试.....	10
2.4.1	实验步骤.....	10
2.4.2	实验调试.....	15
2.4.3	自动化部署脚本.....	16
2.4.4	常见问题.....	18
3	实验三 增加设备驱动程序	20
3.1	实验目的	20
3.2	实验内容	20
3.3	实验设计	20
3.3.1	开发环境.....	20
3.3.2	实验设计.....	20
3.3.3	相关参数.....	21
3.4	实验调试.....	21
3.4.1	实验步骤.....	21
3.4.2	自动化部署.....	24
3.4.3	常见问题.....	25
4	实验四 系统监控器.....	26
4.1	实验目的	26
4.2	实验内容	26
4.3	实验设计	26
4.3.1	开发环境.....	26
4.3.2	实验设计.....	26
4.4	实验调试.....	28

4.4.1	进程信息.....	28
4.4.2	PID 列表扫描的优化	28
4.4.3	数据格式.....	29
4.4.4	单元测试.....	29
4.4.5	GUI 开发	30
4.4.6	集成测试.....	31
4.5	常见问题	31
4.5.1	运行时报进程文件不存在.....	31
4.5.2	关于 proc 文件	31
5	实验五 文件系统.....	33
5.1	实验目的	33
5.2	实验内容	33
5.3	实验设计	33
5.3.1	实验环境.....	33
5.3.2	实验设计.....	33
5.3.3	命令列表.....	33
5.3.4	数据结构.....	34
5.4	技术细节	34
5.4.1	类的继承和调用.....	34
5.4.2	日志记录.....	35
5.4.3	命令解析.....	36
5.4.4	列表的美化输出.....	36
5.4.5	命名检查.....	37
5.4.6	数据的序列化和持久化存储.....	37
5.5	实验测试	38
6	实验总结	41
7	Appendix 实验源码.....	42
7.1	实验一	42
7.1.1	Copy Trigger	42
7.1.2	Multiprocessing Demo	46
7.2	实验二	59
7.2.1	业务代码部分.....	59
7.2.2	部署部分.....	60
7.2.3	利用部分.....	61
7.3	实验三.....	62
7.4	实验四	65
7.5	实验五	75

1 实验一 熟悉和理解 Linux 编程环境

1.1 实验目的

熟悉和理解 Linux 编程环境.

1.2 实验内容

1. 编写一个 C 程序，用 `read`、`write` 等系统调用实现文件拷贝功能。命令形式：
`copy <源文件名> <目标文件名>`
2. 编写一个 C 程序，使用图形编程库 (QT/GTK)分窗口显示三个并发进程的运行(一个窗口实时显示当前系统时间，一个窗口循环显示 0 到 9，一个窗口做 1 到 1000 的累加求和，刷新周期均为 1 秒)。

1.3 实验设计

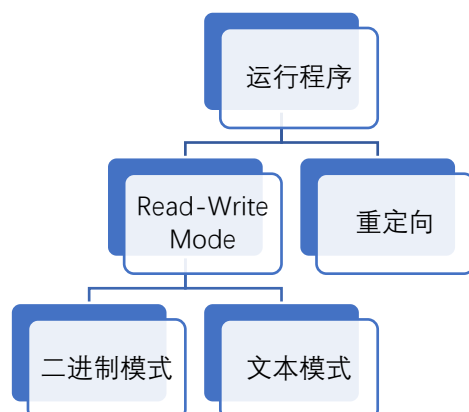
1.3.1 开发环境

Ubuntu 18.04.3 LTS

QT

1.3.2 实验设计

第一部分,设施了两种模式,第一种是使用重定向流作为复制方法,将输入流直接导入到输出流中.第二种方法是使用 C 语言传统的 `read-write`,使用缓冲区进行操作.在运行时使用 Y/N 进行操作.



第二部分,考虑在 `main` 函数中开辟三个子进程(使用 `fork` 语句), 每个进程由两个线程组成, 其中一个线程负责信息的更新,另一个是 UI 的 `mainloop`.三个子进程

分别对应展示时间、0-9 循环和加法。

主入口使用 `fork()` 创造三个进程,分别创建 QT 类 `QApplication`,作为 `Counter`, `Accumulator`, `Clock`。

`Clock` 类使用 `QTimer` 库,每秒获取更新一次时间。

`Counter` 类也使用 `QTimer` 类进行即时,到 10 即约 10,只显示个位数。

`Accumulator` 类是每秒更新,加一个 `Current Num`。

最后三个窗口显示结果和 PID。

1.4 实验调试

1.4.1 实验步骤

1.4.1.1 Copy Trigger

为部署方便,使用了自动化脚本一键编译运行。

```
1. g++ -o copytrigger -g copytrigger.cpp
2. echo compile finished
3. ./copytrigger copy src dst
```

```
compile finished
Confirm:
  Src:src
  Dst:dst
  Size:61
Continue?[Y/n/U(Using read&write)]
y
outfile flished
Process Finished
pota@ubuntu:~/Documents/Copytrigger$ sdiff src dst
Across the Great Wall We Can Reach Everywhere of the World.      Across the Great
Wall We Can Reach Everywhere of the World.
```

图 1-1 使用重定向流模式

```

compile finished
Confirm:
    Src:src
    Dst:dst
    Size:61
Continue?[Y/n/U(Using read&write)]
u
Binary or Document?
[B/D]
d
Confirm:
    Document Mode
Across the Great Wall We Can Reach Everywhere of
the World.

Document Mode Succeed
pota@ubuntu:~/Documents/Copytrigger$ sdiff src dst
Across the Great Wall We Can Reach Everywhere of the World.      Across the Great
Wall We Can Reach Everywhere of the World.

```

图 1-2 使用 Read/Write 模式

1.4.1.2 Multiprocessing Demo

在 QtCreator 中创建工程,导入文件.

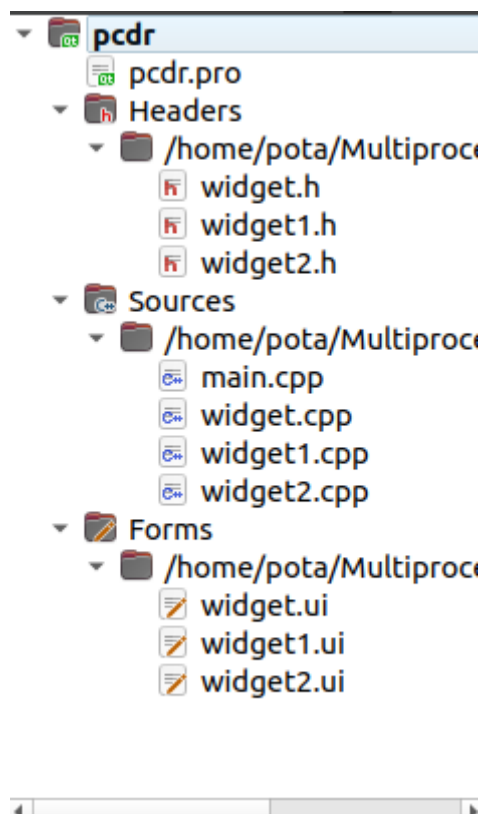


图 1-3

编译运行.

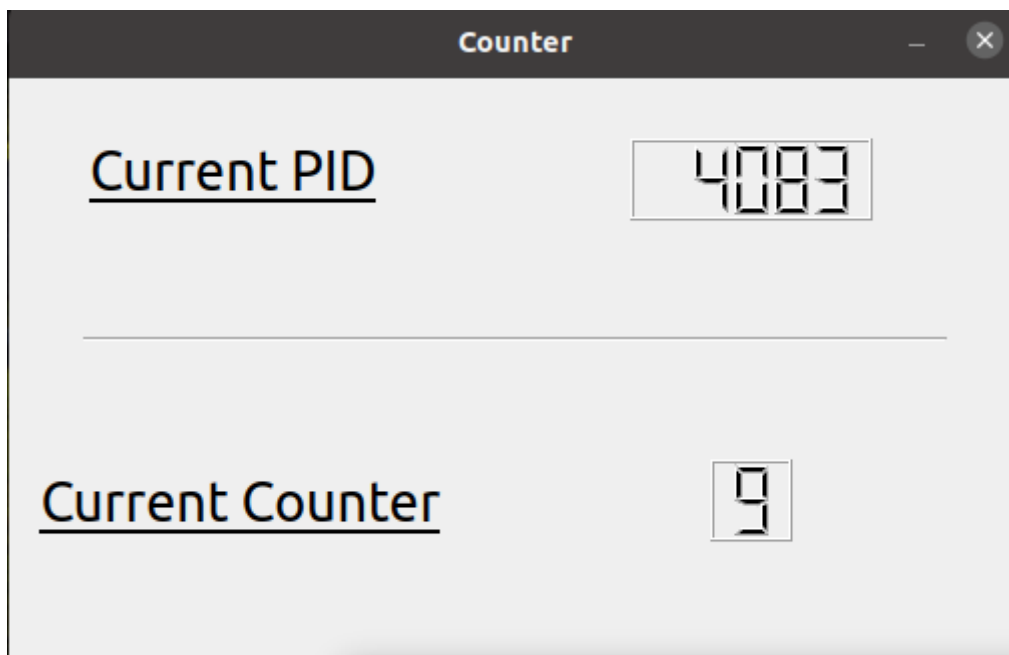


图 1-4 Counter 窗口

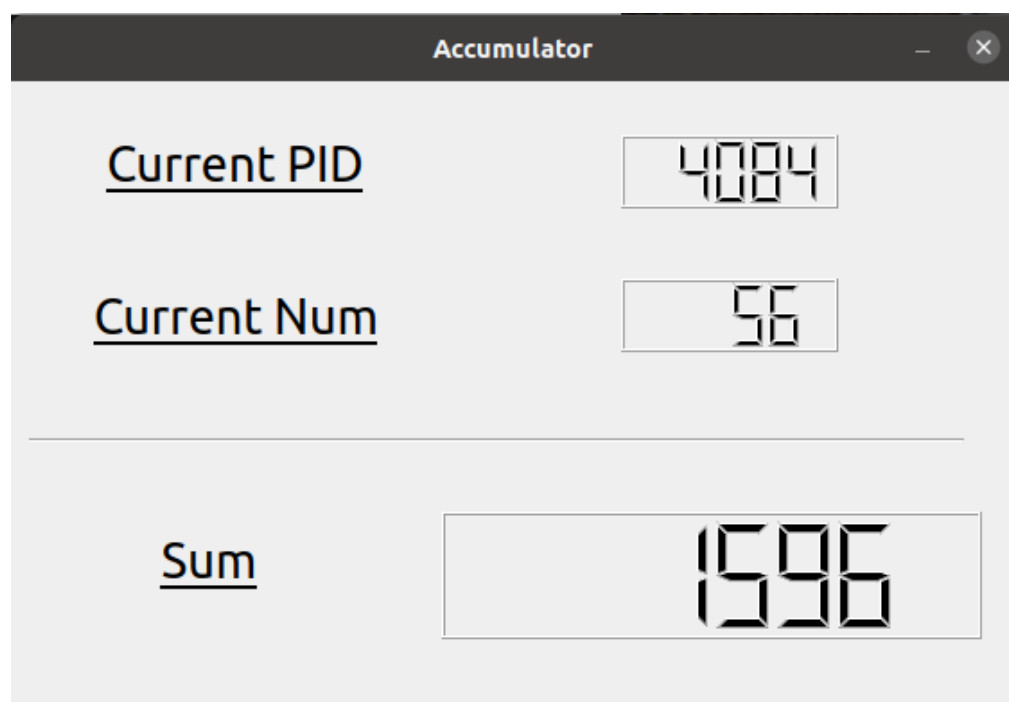


图 1-5 Accumulator 窗口

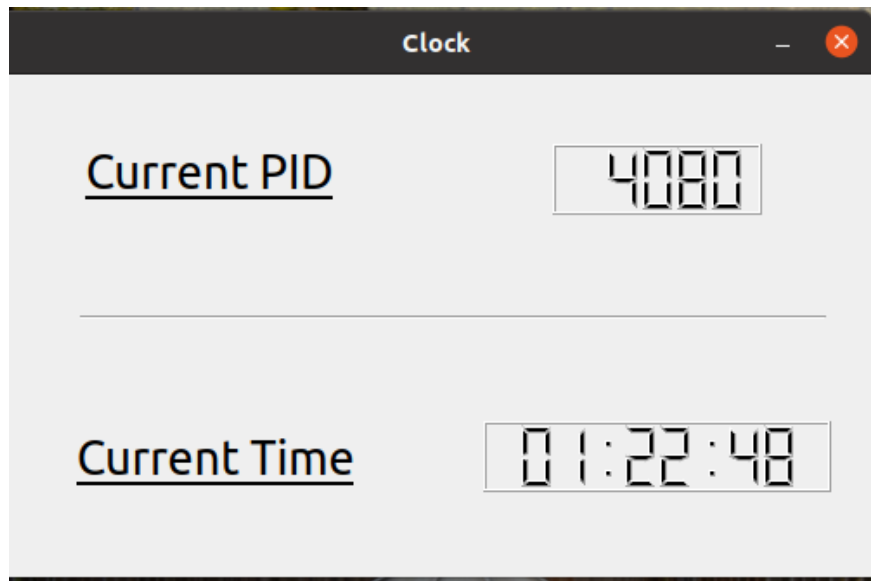


图 1-6 Clock 窗口.由于没有调整时区所以是 GMT+0 时间.

1.4.2 实验调试及心得

1.4.2.1 QT 报 Could not start process "make" qmake all

`sudo apt-get install build-essential`

1.4.2.2 Linux Qt cannot find -lGL

```
1. #查找 libGL 所在位置
2. [root@localhost ~]# locate libGL
3. /usr/lib64/libGL.so
4. /usr/lib64/libGL.so.1
5. /usr/lib64/libGL.so.1.2.0
6. /usr/share/doc/mesa-libGL-9.2.5
7. /usr/share/doc/mesa-libGL-9.2.5/COPYING
8. #创建链接
9. [root@localhost ~]# ln -s /usr/lib64/libGL.so.1 /usr/lib/libGL.so
```

1.4.2.3 GL/gl.h: No such file or directory Or cannot find -lGL

`sudo apt-get install mesa-common-dev`

2 实验二 添加系统调用

2.1 实验目的

熟悉和理解 Linux 编程环境.

2.2 实验内容

1. 采用编译内核的方法，添加一个新的系统调用，实现文件拷贝功能
2. 编写一个应用程序，测试新加的系统调用

2.3 实验设计

2.3.1 开发环境

- Ubuntu 18.04.3 LTS

- Old Kernel: Linux ubuntu 5.0.0-23-generic #24~18.04.1-Ubuntu SMP Mon Jul 29 16:12:28 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux

- New Kernel: Linux ubuntu 4.15.2.POTASSIUM #1 SMP Thu Feb 20 06:51:24 PST 2020 x86_64 x86_64 x86_64 GNU/Linux

2.3.2 实验设计

Linux 内核中设置了一组用于实现各种系统功能的子程序，称为系统调用。用户可以通过系统调用命令在自己的应用程序中调用它们。从某种角度来看，系统调用和普通的函数调用非常相似。区别仅仅在令在自己的应用程序中调用它们。从某种角度来看，系统调用和普通的函数调用非常相似。区别仅仅在于，系统调用由操作系统核心提供，运行于核心态；而普通的函数调用由函数库或用户自己提供，运行于用户态。二者在使用方式上也有相似之处。Linux 系统的核心部分即是 Linux 内核，是一系列设备的驱动程序。系统调用是 Linux 内核提供的功能十分强大的一系列的函数。这些函数是在内核中实现的，它们是应用程序和内核交互的接口，系统调用在 Linux 系统中发挥着巨大的作用，如果没有系统调用，那么应用程序就失去了内核的支持。

2.4 实验调试

2.4.1 实验步骤

2.4.1.1 编译原内核

2.4.1.1.1 环境要求

- gcc
- make
- build-essential
- ncurses-dev
- libssl-dev

2.4.1.1.2 编译步骤

为了修改方便设置了自动编译脚本.(make menuconfig 生成内核配置文件已经做过)

```
1. cd ..  
2. cd linux-4.15.2  
3. echo "Start"  
4. sudo make clean  
5. sudo make -j4 bzImage  
6. echo "Image Compile Finished"  
7. sudo make -j4 modules  
8. echo "Modules Finished"  
9. sudo make -j4 modules_install  
10. echo "Module Installed"  
11. sudo make -j4 install  
12. echo "Finished"
```

1. 进入内核目录
2. 编译 Image
3. 编译 Module
4. 安装 Module
5. 安装内核

2.4.1.1.3 切换内核

1. 设置自动登录防止登录校验阶段无法进入系统.
2. 重启,进入 GNU GRUB 列表,选择 Advanced Options

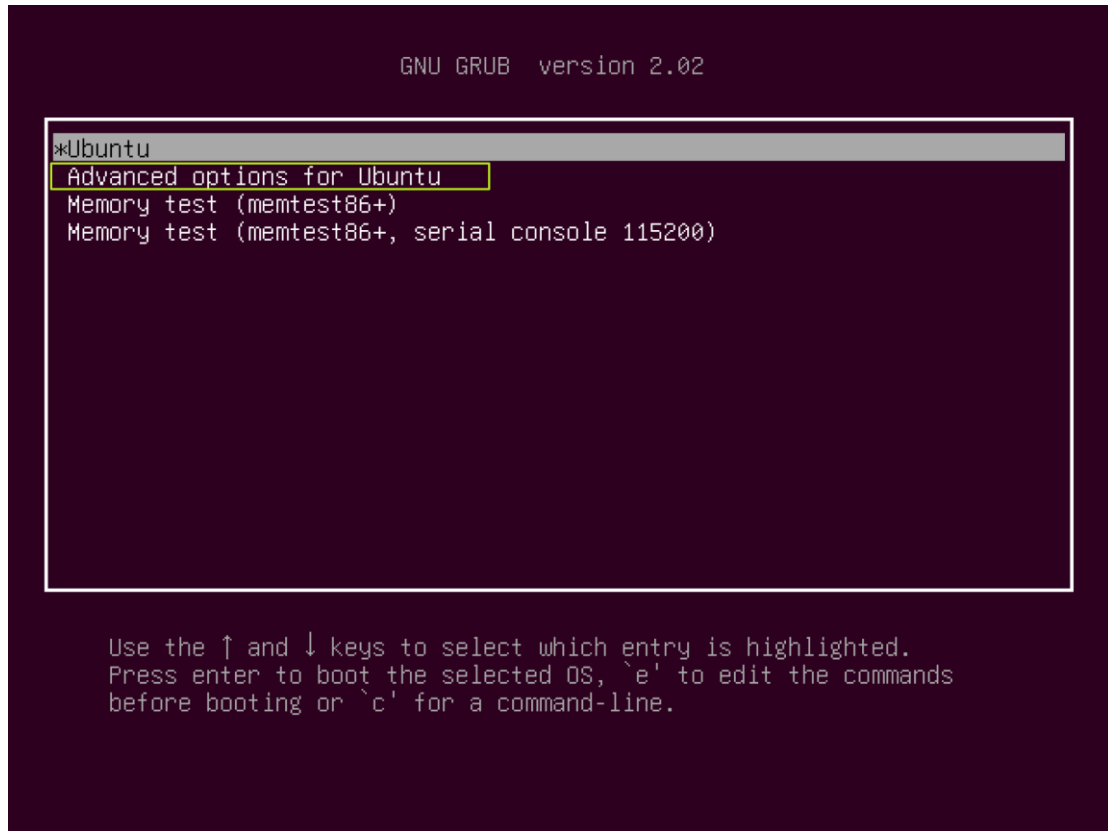


图 2-1

3. 选择 4.15.2.POTASSIUM(为了分辨方便加上了别名 POTASSIUM)
4. 进入新内核.

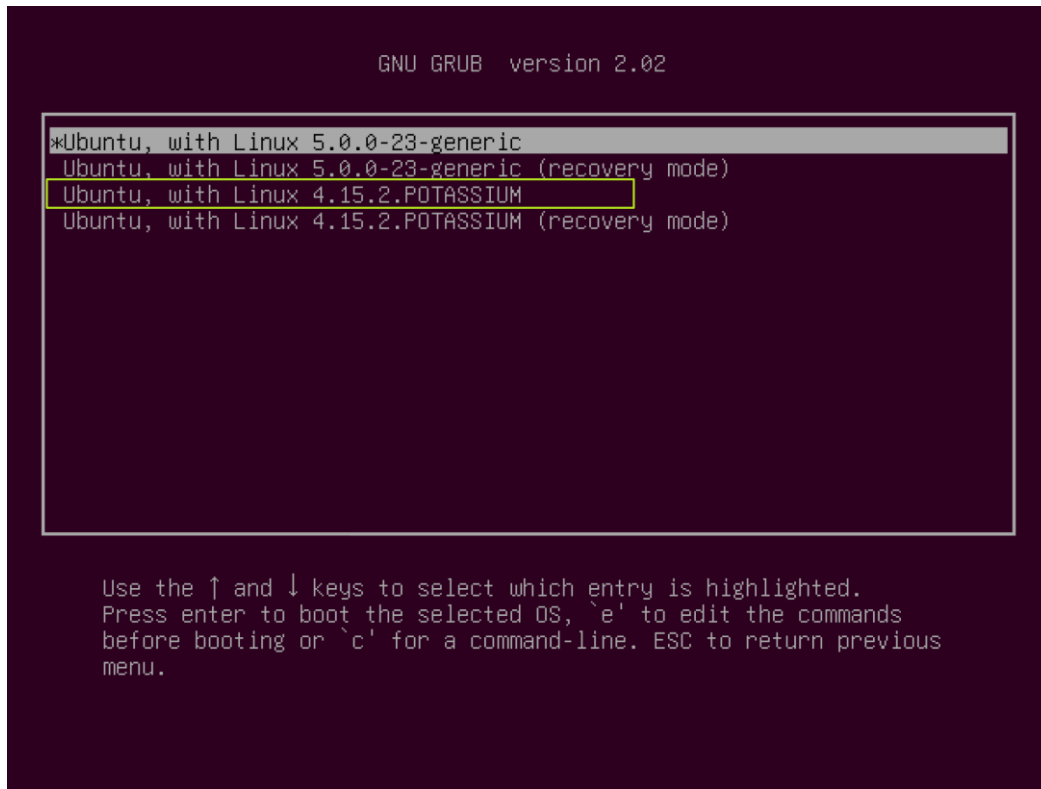


图 2-2

2.4.1.2 添加新的系统调用

2.4.1.2.1 代码生成

在/usr/src/linux4-14.123/kernel/sys.c 中添加 sys_customcp。

```
1. asmlinkage int sys_customcp(const char *src, const char *dst)
2. {
3.     int infd, outfd, count;
4.     char buf[256];
5.     mm_segment_t fs;
6.     fs = get_fs();
7.     set_fs(get_ds( ));
8.     //Handle Error
9.     if((infd=sys_open(src, O_RDONLY, 0)) == -1)
10.    {
11.        return 1;
12.    }
13.    if((outfd=sys_open(dst, O_WRONLY | O_CREAT, S_IRUSR| S_IWUSR)) ==-1)
14.    {
15.        return 2;
```

```

16.     }
17.     while((count = sys_read(infd, buf, 256)) > 0)
18.     {
19.         if(sys_write(outfd, buf, count) != count)
20.             return 3;
21.     }
22.     if(count == -1)
23.         return 4;
24.     sys_close(infd);
25.     sys_close(outfd);
26.     set_fs(fs);
27.     return 0;
28. }

```

并在 `./include/linux/syscalls.h` 中添加声明 `asmlinkage int sys_customcp(const char *src, const char *dst);`

在 `/usr/src/linux-4.14.123/arch/x86/entry/syscalls/syscall_64.tbl` 添加系统调用号.

```

1. 333 64 customcp sys_customcp

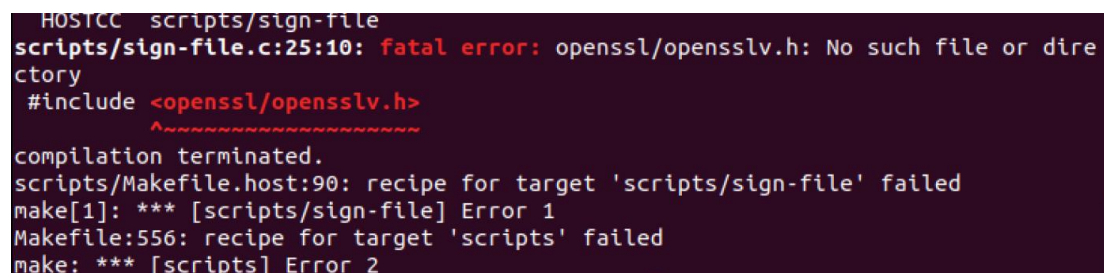
```

2.4.1.2.2 环境准备

1. gcc
2. make
3. build-essential (include g++ 和 Linux-kernel-headers)
4. 用 `make menuconfig` 设置核心参数需先安装 `ncurses`
5. `sudo apt-get install ncurses-dev`

使用 `sudo make menuconfig` 准备配置. 如果出现下面错误则需要安装 `libssl`:

`sudo apt-get install libssl-dev`



```

HOSTCC scripts/sign-file
scripts/sign-file.c:25:10: fatal error: openssl/opensslv.h: No such file or directory
#include <openssl/opensslv.h>
compilation terminated.
scripts/Makefile.host:90: recipe for target 'scripts/sign-file' failed
make[1]: *** [scripts/sign-file] Error 1
Makefile:556: recipe for target 'scripts' failed
make: *** [scripts] Error 2

```

图 2-3

2.4.1.2.3 编译

```
LD [M] sound/soundcore.ko
CC sound/synth/emux/snd-emux-synth.mod.o
LD [M] sound/synth/emux/snd-emux-synth.ko
CC sound/synth/snd-util-mem.mod.o
LD [M] sound/synth/snd-util-mem.ko
CC sound/usb/6fire/snd-usb-6fire.mod.o
LD [M] sound/usb/6fire/snd-usb-6fire.ko
CC sound/usb/bcd2000/snd-bcd2000.mod.o
LD [M] sound/usb/bcd2000/snd-bcd2000.ko
CC sound/usb/caiaq/snd-usb-caiaq.mod.o
LD [M] sound/usb/caiaq/snd-usb-caiaq.ko
CC sound/usb/hiface/snd-usb-hiface.mod.o
LD [M] sound/usb/hiface/snd-usb-hiface.ko
CC sound/usb/line6/snd-usb-line6.mod.o
LD [M] sound/usb/line6/snd-usb-line6.ko
CC sound/usb/line6/snd-usb-pod.mod.o
LD [M] sound/usb/line6/snd-usb-pod.ko
CC sound/usb/line6/snd-usb-podhd.mod.o
LD [M] sound/usb/line6/snd-usb-podhd.ko
CC sound/usb/line6/snd-usb-toneport.mod.o
LD [M] sound/usb/line6/snd-usb-toneport.ko
CC sound/usb/line6/snd-usb-variiax.mod.o
LD [M] sound/usb/line6/snd-usb-variiax.ko
CC sound/usb/misc/snd-ua101.mod.o
LD [M] sound/usb/misc/snd-ua101.ko
CC sound/usb/snd-usb-audio.mod.o
LD [M] sound/usb/snd-usb-audio.ko
CC sound/usb/snd-usbmidi-lib.mod.o
LD [M] sound/usb/snd-usbmidi-lib.ko
CC sound/usb/usx2y/snd-usb-us122l.mod.o
LD [M] sound/usb/usx2y/snd-usb-us122l.ko
CC sound/usb/usx2y/snd-usb-usx2y.mod.o
LD [M] sound/usb/usx2y/snd-usb-usx2y.ko
CC sound/x86/snd-hdmi-lpe-audio.mod.o
LD [M] sound/x86/snd-hdmi-lpe-audio.ko
CC virt/lib/irqbypass.mod.o
LD [M] virt/lib/irqbypass.ko
```

图 2-4

重启,切换内核进入系统.

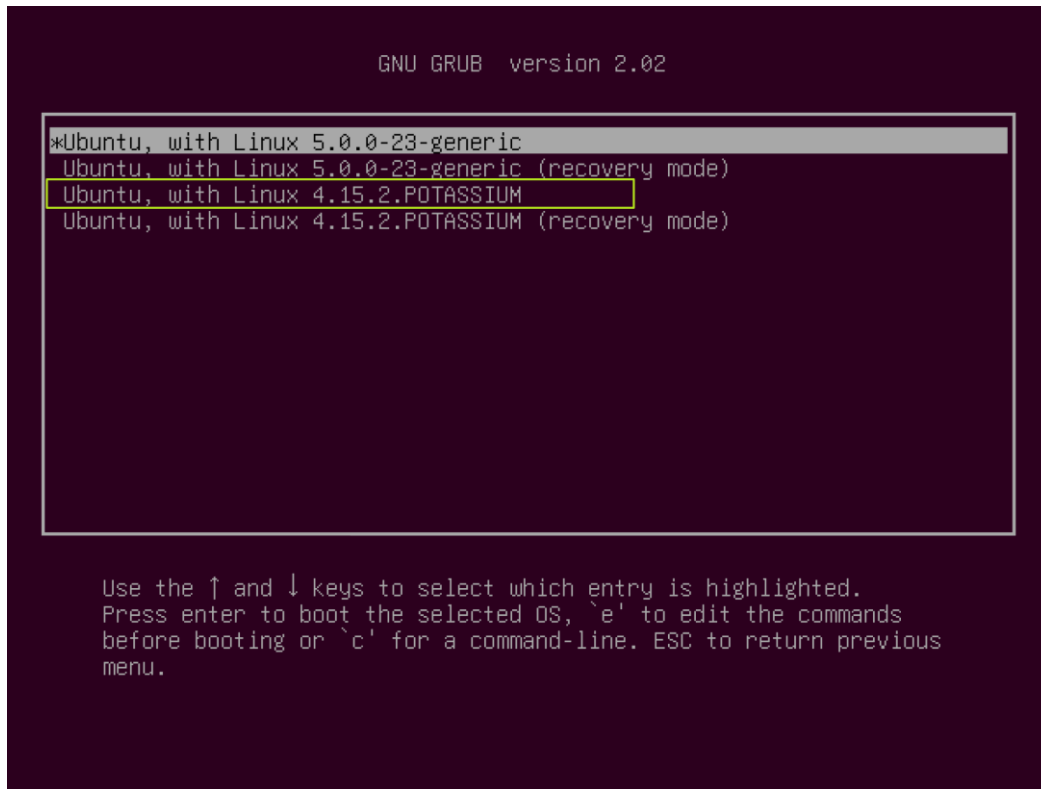


图 2-5

使用该调用测试复制.

```
sysproj@ubuntu:~/Documents/AutoDeploy/Test Sample$ ./shellcode copy src dst
Confirm:copy src dst
Success!
```

图 2-6

测试成功.

```
sysproj@ubuntu:~/Documents/AutoDeploy/Test Sample$ sdiff src dst
Across the Great Wall we can reach everywhere of the world. Across the Great Wall we can reach everywhere of the world.
```

图 2-7

2.4.2 实验调试

2.4.2.1 系统调用利用代码

```
1. #include <stdio.h>
2. #include <linux/kernel.h>
3. #include <unistd.h>
4. #include <string.h>
5. int main(int argc, char* argv[])
```

```

6. {
7.     if (argc!=3||strcmp(argv[0],"copy"))
8.     {
9.         printf("Usage: copy <src> <dst>");
10.        return 0;
11.    }
12.    else
13.    {
14.        printf("Comfirm:%s %s %s\n",argv[0],argv[1],argv[2]);
15.        int status=syscall(333,argv[1],argv[2]);
16.        if (status==1)
17.        {
18.            printf("Failed: Unable to open arc file\n");
19.        }
20.        else if (status==2)
21.        {
22.            printf("Failed: Unable to open dst file\n");
23.        }
24.        else if (status==3)
25.        {
26.            printf("Failed: Buffer Overflow\n");
27.        }
28.        else if (status==4)
29.        {
30.            printf("Failed: Error while cpying\n");
31.        }
32.        else if (!status)
33.        {
34.            printf("Success!\n");
35.        }
36.        return 0;
37.    }
38. }

```

使用参数 Usage: copy <src> <dst>

2.4.3 自动化部署脚本

由于改动了很多次,为保证效率及避免重复工作,使用自动部署脚本.

1. 给两个部署脚本可执行权限.
2. 运行 autodeploy.sh
3. 手动生成 menuconfig
 - a) sudo make menuconfig
 - b) 保存退出

4. 检查配置,运行 autocompile.sh
5. 开始编译.

2.4.3.1 Auto Deploy.sh

```
1. # Env Check
2. sudo apt-get update
3. sudo apt-get install gcc
4. sudo apt-get install make
5. sudo apt-get install build-essential
6. sudo apt-get install ncurses-dev
7. sudo apt-get install libssl-dev
8. echo "Environment Check Finished..."
9.
10. # Deploy
11. sudo cp -fp syscall_64.tbl ../linux-4.15.2/arch/x86/entry/syscalls/
12. sudo cp -fp syscalls.h ../linux-4.15.2/include/linux/
13. sudo cp -fp sys.c ../linux-4.15.2/kernel/
14. echo "Finished"
```

1. 检查环境配置,更新依赖
2. 将修改后的相关文件覆盖原文件

2.4.3.2 Autocompile.sh

```
1. cd ..
2. cd linux-4.15.2
3. echo "Start"
4. sudo make clean
5. sudo make -j4 bzImage
6. echo "Image Compile Finished"
7. sudo make -j4 modules
8. echo "Modules Finished"
9. sudo make -j4 modules_install
10. echo "Module Installed"
11. sudo make -j4 install
12. echo "Finished"
```

1. 清除之前编译痕迹
2. 编译 Image
3. 编译 Module
4. 安装 Module

5. 总安装

2.4.3.3 Makefile

```
1. # SPDX-License-Identifier: GPL-2.0
2. VERSION = 4
3. PATCHLEVEL = 15
4. SUBLEVEL = 2
5. EXTRAVERSION = .POTASSIUM
6. NAME = Fearless Coyote
```

EXTRAVERSION 即为识别方便添加的 Alias.

2.4.4 常见问题

2.4.4.1 编译速度过慢

在虚拟机使用多核情况下可以考虑使用多线程编译. 这样设置后 CPU 的利用率可以达到 90%左右.

2.4.4.1.1 无法进入 GRUB

1. 将 etc/default/grub 设置为可写文件:
2. `sudo chmod 666 grub`
3. 修改文件: ~~GRUB_TIMEOUT=0~~ GRUB_TIMEOUT=20 或者更大值
4. 为了方便定位错误,修改: ~~GRUB_CMDLINE_LINUX_DEFAULT="quiet"~~
GRUB_CMDLINE_LINUX_DEFAULT="text"
5. 保存退出
6. 更新 Grub 设置: `sudo update-grub`

2.4.4.2 进入 GRUB 前卡死

可能是没有关闭 User Manager,即每次开机的登录界面.
关闭方法:

1. 打开右上角的设置
2. 点击 system settings
3. 点击 user accounts
4. 点击右上角的 Unlock

5. 输入密码
6. 最后将 Automatic login 设置为 on 打开
7. 最后点击后上角的 lock，锁定设置

2.4.4.3 切换回原来的内核

如果要换回原来的内核，则先使用命令

```
grep menuentry /boot/grub/grub.cfg
```

该命令显示内核的顺序.假设你要以某内核版本启动,则将文件/etc/default/grub 中 GRUB_DEFAULT=0 改为 GRUB_DEFAULT=\$列表中的序号.保存后使用命令 sudo update-grub 更新 GRUB.

重启后，使用命令 uname -a 查看，内核即为原来的内核.

3 实验三 增加设备驱动程序

3.1 实验目的

掌握添加设备驱动程序的方法

3.2 实验内容

1. 采用模块方法，添加一个新的字符设备驱动程序，实现打开/关闭、读/写等基本操作
2. 编写一个应用程序，测试添加的驱动程序

3.3 实验设计

3.3.1 开发环境

Ubuntu 18.04.3 LTS

Kernel: Linux ubuntu 5.0.0-23-generic

3.3.2 实验设计

linux 系统将设备分为 3 类：字符设备、块设备、网络设备。

字符设备

字符设备是能够像字节流(类似文件)一样被访问的设备，有字符设备驱动程序来实现这种特性。字符设备驱动程序通常至少要实现 `open`、`close`、`read`、`write` 系统调用。字符设备可以通过文件系统节点来访问，这些设备文件和普通文件之间的唯一差别在于对普通文件的访问可以前后移动访问位置，而大多数字符设备是一个只能顺序访问的数据通道。一个字符设备是一种字节流设备，对设备的存取只能按顺序按字节的存取而不能随机访问，字符设备没有请求缓冲区，所有的访问请求都是按顺序执行的。但事实上现在一些高级字符设备也可以从指定位置一次读取一块数据。

块设备

块设备也是通过设备节点来访问。块设备上能够容纳文件系统。在大多数 unix 系统中，进行 I/O 操作时块设备每次只能传输一个或多个完整的块，而每块包含 512 字节（或更 2 的更高次幂字节的数据）。linux 可以让应用程序向字符设备一样读

写块设备，允许一次传递任意多字节的数据。因而，块设备和字符设备的区别仅仅在于内核内部管理数据的方式，也就是内核及驱动程序之间的软件接口，而这些不同对用户来讲是透明的。在内核中，和字符驱动程序相比，块驱动程序具有完全不同的接口。存储设备一般属于块设备，块设备有请求缓冲区，并且支持随机访问而不必按照顺序去存取数据，比如你可以先存取后面的数据，然后在存取前面的数据，这对字符设备来说是不可能的。Linux 下的磁盘设备都是块设备，尽管在 Linux 下有块设备节点，但应用程序一般是通过文件系统及其高速缓存来访问块设备的，而不是直接通过设备节点来读写块设备上的数据。

网络设备

网络设备不同于字符设备和块设备，它是面向报文的而不是面向流的，它不支持随机访问，也没有请求缓冲区。由于不是面向流的设备，因此将网络接口映射到文件系统中的节点比较困难。内核和网络设备驱动程序间的通讯，完全不同于内核和字符以及块驱动程序之间的通讯，内核调用一套和数据包传输相关的函数而不是 `read`, `write`。网络接口没有像字符设备和块设备一样的设备号，只有一个唯一的名字，如 `eth0`、`eth1` 等，而这个名字也不需要与设备文件节点对应。

本次实验使用的是字节流设备。

3.3.3 相关参数

设备名: `customdev`

设备号: 一般为 240

Banner: `customdev by POTASSIUM`

3.4 实验调试

3.4.1 实验步骤

3.4.1.1 编写设备

1) 注册模块

`register_chrdev (unsigned int major, const char *name, struct file_operations *fops);` 用于向系统的字符设备表登记一个字符设备，成功返回设备的主设备号，否则返回一个负值。

2) 注销模块

使用 `unregister_chrdev (unsigned int major, const char *name);` 注销字符设备。

3) 打开

使用 `try_module_get(THIS_MODULE);` 函数，判断 `module` 模块是否处于活动状态，将该模块的引用计数加 1。

读取写入设备里的文件，`copy_to_user(void __user *to, const void *from, unsigned long n)` 和 `copy_from_user(void *to, const void __user *from, unsigned long n)`。这两个函数是作用于内核函数，而不是指向用户的函数。

释放模块,使用 `module_put(THIS_MODULE);`函数,使指定的模块使用量-1,拥有该结构的模块的指针设备驱动模块编译.

3.4.1.2 部署设备

准备 Makefile 文件,使用 `sudo` 编译.

```
pota@ubuntu:~/Documents/eck$ sudo make
make -C /lib/modules/5.4.0-37-generic/build M=/home/pota/Documents/eck
make[1]: 进入目录“/usr/src/linux-headers-5.4.0-37-generic”
  AR      /home/pota/Documents/eck/built-in.a
  CC [M]  /home/pota/Documents/eck/basindex.o
Building modules, stage 2.
MODPOST 1 modules
  CC [M]  /home/pota/Documents/eck/basindex.mod.o
  LD [M]  /home/pota/Documents/eck/basindex.ko
make[1]: 离开目录“/usr/src/linux-headers-5.4.0-37-generic”
```

图 3-1

加载设备驱动模块

1. `sudo insmod basindex.ko`

之后可以在 `/proc/devices` 中找到设备名和设备号.

```
pota@ubuntu:~/Documents/eck$ cat /proc/devices|grep custom
240 customdev
```

图 3-2

注册设备,注册完成后可以在 `/dev` 目录下找到相关设备文件

```
pota@ubuntu:/dev$ sudo mknod /dev/customdev c 240 0
pota@ubuntu:/dev$ find customdev
customdev
pota@ubuntu:/dev$ ls | grep customdev
customdev
```

图 3-3

3.4.1.3 设备测试

编写测试代码.

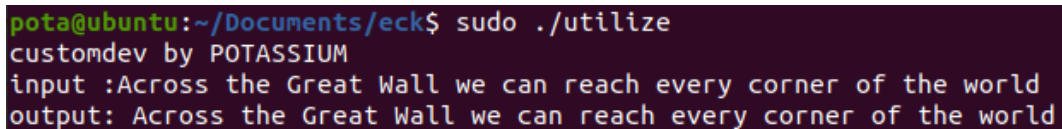
```
1. #include <sys/types.h>
2. #include <sys/stat.h>
3. #include <stdlib.h>
4. #include <string.h>
5. #include <stdio.h>
6. #include <fcntl.h>
7. #include <unistd.h>
8. #define MAX_SIZE 1024
```

```

9.
10. int main(void)
11. {
12.     int customdev;
13.     char buf[MAX_SIZE];    //缓冲区
14.     char get[MAX_SIZE];    //要写入的信息
15.
16.     char dir[50] = "/dev/customdev";    //设备名
17.
18.     customdev = open(dir, O_RDWR | O_NONBLOCK);
19.     if (customdev==-1)
20.     {
21.         printf("Cann't open file \n"); exit(0);
22.         return -1;
23.     }
24.     //读初始信息
25.     read(customdev, buf, sizeof(buf));
26.     printf("%s\n", buf);
27.
28.     //写信息
29.     printf("input :");
30.     gets(get);
31.     write(customdev, get, sizeof(get));
32.
33.     //读信息
34.     read(customdev, buf, sizeof(buf));
35.     printf("output: %s\n", buf);
36.
37.     close(customdev);
38.     return 0;
39.
40. }

```

编译,运行测试.



```

pota@ubuntu:~/Documents/eck$ sudo ./utilize
customdev by POTASSIUM
input :Across the Great Wall we can reach every corner of the world
output: Across the Great Wall we can reach every corner of the world

```

图 3-4

测试成功.

3.4.1.4 卸载设备

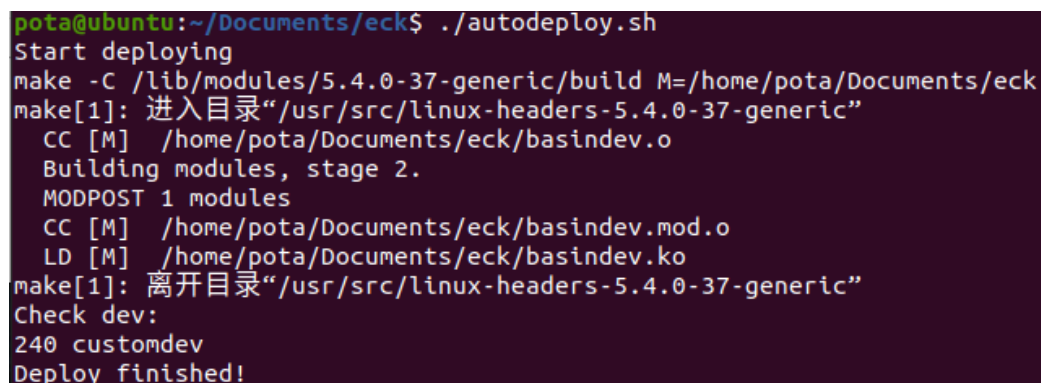
```
1. sudo make clean
2. sudo rmmod customdev
3. sudo rm -r /dev/customdev
```

卸载清除完毕.

3.4.2 自动化部署

考虑到实验会因为很多细节失败,而彻底清除之前的残留非常困难.这里考虑使用自动化脚本辅助部署.

```
1. echo "Start deploying";
2. sudo make;
3. sudo insmod basindeb.ko;
4. echo "Check dev\:";
5. sudo mknod /dev/customdev c 240 0;
6. sudo lsmod;
7. echo "Deploy finished\!";
```



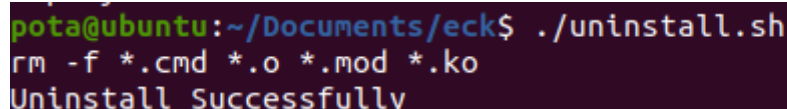
```
pota@ubuntu:~/Documents/eck$ ./autodeploy.sh
Start deploying
make -C /lib/modules/5.4.0-37-generic/build M=/home/pota/Documents/eck
make[1]: 进入目录"/usr/src/linux-headers-5.4.0-37-generic"
CC [M] /home/pota/Documents/eck/basindeb.o
Building modules, stage 2.
MODPOST 1 modules
CC [M] /home/pota/Documents/eck/basindeb.mod.o
LD [M] /home/pota/Documents/eck/basindeb.ko
make[1]: 离开目录"/usr/src/linux-headers-5.4.0-37-generic"
Check dev:
240 customdev
Deploy finished!
```

图 3-5

部署成功.

卸载脚本:

```
1. sudo make clean;
2. sudo rmmod basindeb;
3. sudo rm -r /dev/customdev;
4. echo "Uninstall Successfully";
```



```
pota@ubuntu:~/Documents/eck$ ./uninstall.sh
rm -f *.cmd *.o *.mod *.ko
Uninstall Successfully
```

图 3-6

这样可以成功清除所有残留.

3.4.3 常见问题

3.4.3.1 CONFIG_X86_X32 enabled but no binutils support

可能是跨内核编译的后果.考虑重新编译而不是使用原来的二进制文件.

3.4.3.2 rmmod: ERROR: Module customdev is not currently loaded

卸载的时候出现的问题.

需要注意这里的参数是模块名(即*.ko).如果模块名和设备名不同则容易出现这样的错误.当然如果没有填写正确模块名也会报这个错因为找不到相关模块.

3.4.3.3 \$'\r': 未找到

这个出现在脚本编写的过程中.

一般是从 Windows 的 Dos 系统转到 Linux 的 Unix 系统时出现的问题.可以使用 dos2unix 解决.

应该是两种操作系统换行符的不同造成的问题.Dos 是 CR LF.

Sudo apt-get install dos2unix

Dos2unix <Filename>

3.4.3.4 ***没有规则可以创建“XXX”

Makefile 首字母要大写.

4 实验四 系统监控器

4.1 实验目的

1. 了解/proc 文件的特点和使用方法
2. 监控系统状态，显示系统部件的使用情况
3. 用图形界面监控系统状态，包括 CPU 和内存利用率、所有进程信息等
(可自己补充、添加其他功能)

4.2 实验内容

实现一个系统监控器，具有以下监控系统功能

通过读取 proc 文件系统，获取系统各种信息，并以比较容易理解的方式显示出来.

4.3 实验设计

4.3.1 开发环境

Ubuntu 20.04 LTS

Linux version: 5.4.0-37-generic

Python 3.8.2

4.3.2 实验设计

考虑使用 tkinter in python3 实现 GUI.

显示的信息包括 CPU 信息,内存信息,操作系统信息,进程信息.

原型图设计如下:

sys monitor						×
CPU信息	PID	NAME	PPID	STATUS	THREADS	
内存信息						
操作系统信息						

图 4-1

4.3.2.1 数据结构

- 1. PID information
 - 1. PID /\$pid
 - 2. Name /status/Name
 - 3. Parent PID /status/PPid
 - 4. State /status/State
 - 5. Threads /status/Threads
- 2. CPU information
 - 1. 生产商 vendor_id
 - 2. 系列代号 cpu family
 - 3. 名称 model name
 - 4. 频率 cpu MHz
 - 5. 核数 cpu cores
- 3. Memory information
 - 1. 总内存 Memtotal
 - 2. 空闲内存 Memfree
- 4. System information

4.3.2.2 文件结构

main.py: 主入口和主循环
 tk.py: 基本的 Tkinter 组件
 multibox.py:列表组件类
 info.py: 通过\ proc 获取和处理信息

4.4 实验调试

4.4.1 进程信息

4.4.1.1 获取 PID

- 1.使用`ls /proc -F | grep [0-9]*`进行原始测试。 使用绝对路径来确保它可以在任何地方运行
- 2.使用正则表达式匹配: `findall (“ [0-9]+”, str (raw))`
- 3.获取所有 PID 的列表。

4.4.1.2 获取状态信息

访问`/status`以获取信息。

4.4.2 PID 列表扫描的优化

使用改进的解决方案来加速扫描,不需要完整扫描一遍获取的文本就可以解析并结构化所有信息.

```
1. res=["" for _ in range(4)]
2.
3. itemlist=["Name","PPid","State","Threads"]
4.
5. for item in raw:
6.     if itemlist==["" for _ in range(4)]:
7.         break#Finish
8.     else:
9.         for opt in itemlist:
10.            if opt=="":
11.                continue
12.            if item.find(opt)!=-1:#Match
13.                res[itemlist.index(opt)]=item
14.                itemlist[itemlist.index(opt)]=""#Remove opt
15.                break
16. print(res)# Revised: Use dict{name:value} instead.
```

程序执行时,某些进程被杀死。 要从列表中删除它们以避免未知错误.

4.4.3 数据格式

使用字典在数据获取端和 GUI 显示端传递数据.这是一种类 json 的数据格式.数据格式如下:

```
1. {
2.     pidinfo:{PID:[Name,Parent PID,State,Threads]}
3.     cpuinfo:{vendor,family,model,frequency,cores},
4.     meminfo:[total,free],
5.     sysinfo:str
6. }
```

4.4.4 单元测试

在没有 GUI 的情况下测试数据格式和获取情况:

```
1. 获取结果如下:
   {'cpuinfo': {'cpuMHz': ['2808.004'],
2.             'cpucores': ['1'],
3.             'cpufamily': ['6'],
4.             'model': ['158'],
5.             'vendor_id': ['GenuineIntel']}},
6. 'meminfo': {'MemFree': ['172772kB'], 'MemTotal': ['2006888kB']},
7. 'pidinfo': {'1': {'Name': ['systemd'],
8.                  'PPid': ['0'],
9.                  'State': ['S(sleeping)'],
10.                 'Threads': ['1']},
11.            '10': {'Name': ['ksoftirqd/0'],
12.                  'PPid': ['2'],
13.                  'State': ['S(sleeping)'],
14.                  'Threads': ['1']},
15.            '1026': {'Name': ['upowerd'],
16.                   'PPid': ['1'],
17.                   'State': ['S(sleeping)'],
18.                   'Threads': ['3']},
19.            ... ..
20.            },
21. 'sysinfo': 'Linux version 5.4.0-26-generic (buildd@lcy01-amd64-029) (gcc '
22.            'version 9.3.0 (Ubuntu 9.3.0-10ubuntu2)) #30-
23.            Ubuntu SMP Mon Apr 20 '
24.            '16:58:30 UTC 2020\n'
25. }
```

测试结果没有问题.

4.4.5 GUI 开发

根据设计安排每个框架的格式.
每个部分由一个 Frame 控件处理.
Frame 结构如下:

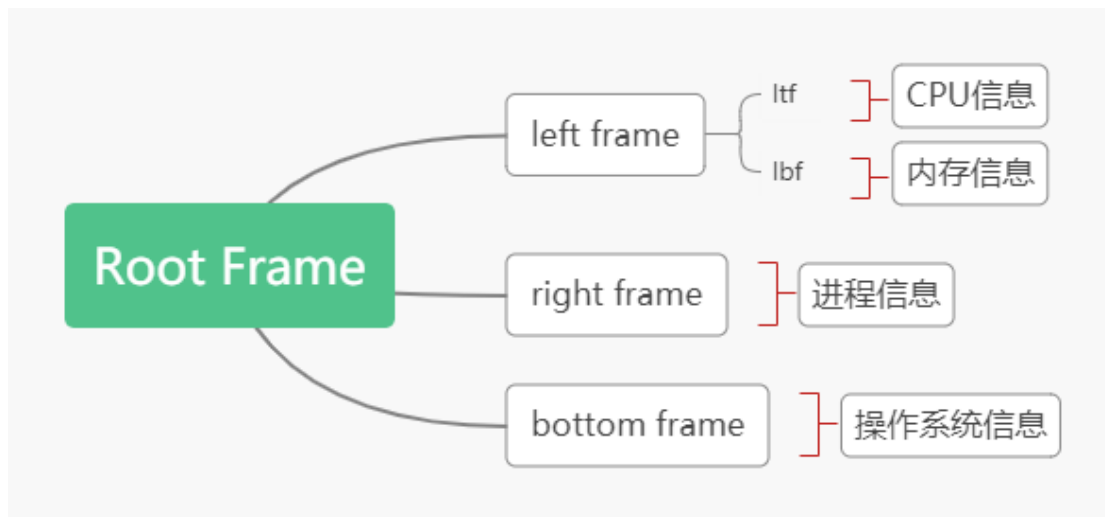


图 4-2

PID 信息,CPU 信息,内存信息使用多栏列表控件 Multi list box.

数据格式为 (((Names), (Value1), (vaule2), ...)

1.将数据 self.procinfo 整理为目标格式.

2.打包时, 使用 “ info [0]” 作为列.

数据整理代码如下:

```
1. self.cpubox=MultiListbox(self.ltf,self.cpuinfo[0])
2. for item in self.cpuinfo[1:]:
3.     self.cpubox.insert(END,item)
4. self.cpubox.pack(expand=YES, fill=BOTH)
5.
6. self.pidbox=MultiListbox(self.rtf,self.pidinfo[0])
7. for item in self.pidinfo[1:]:
8.     self.pidbox.insert(END,item)
9. self.pidbox.pack(expand=YES, fill=BOTH)
10.
11. self.membox=MultiListbox(self.lbf,self.meminfo[0])
12. for item in self.meminfo[1:]:
13.     self.membox.insert(END,item)
14. self.membox.pack(expand=YES, fill=BOTH)
```

4.4.6 集成测试

需要先安装 tkinter 库。
运行入口文件 main.py

System monitor										
vendor_id	cpufamily	model	cpuMHz	cpucores	PID	Name	PPid	State	Threads	
GenuineIntel	6	158	2808.004	1	1	systemd	0	S(sleeping)	1	
					2	kthreadd	0	S(sleeping)	1	
					3	rcu_gp	2	I(idle)	1	
					4	rcu_par_gp	2	I(idle)	1	
					6	kworker/0	2	I(idle)	1	
					1032	rtkit-daemon	1	S(sleeping)	3	
					9	mm_percpu_wq	2	I(idle)	1	
					10	ksoftirqd/0	2	S(sleeping)	1	
					11	rcu_sched	2	I(idle)	1	
					12	migration/0	2	S(sleeping)	1	
					13	idle_inject/0	2	S(sleeping)	1	
					14	cpuhp/0	2	S(sleeping)	1	
					MemFree		MemTotal			
204636kB		2006884kB								
					15	kdevtmpfs	2	S(sleeping)	1	
					16	netns	2	I(idle)	1	
					17	rcu_tasks_kthre	2	S(sleeping)	1	
					18	kauditd	2	S(sleeping)	1	
					19	khungtaskd	2	S(sleeping)	1	
					20	oom_reaper	2	S(sleeping)	1	
					21	writeback	2	I(idle)	1	
					22	kcompactd0	2	S(sleeping)	1	
					23	ksmd	2	S(sleeping)	1	
Linux version 5.4.0-37-generic (buildd@lcy01-amd64-001) (gcc version 9.3.0 (Ubuntu 9.3.0-10ubuntu2)) #41-Ubuntu SMP Wed Jun 3 18:57:02 UTC 2020										

图 4-3

运行成品.较好的反映了查询到的信息.

4.5 常见问题

4.5.1 运行时报进程文件不存在

某些进程在获取 PID 之后被杀死,进程文件被移除,所以找不到.

4.5.2 关于 proc 文件

/proc 文件系统是一个虚拟文件系统,通过它可以使使用一种新的方法在 Linux 内核空间和用户间之间进行通信.在 /proc 文件系统中,我们可以将对虚拟文件的读写作为与内核中实体进行通信的一种手段,但是与普通文件不同的是,这些虚拟文件的内容都是动态创建的.

最初开发 /proc 文件系统是为了提供有关系统中进程的信息.但是由于这个文件系统非常有用,因此内核中的很多元素也开始使用它来报告信息,或启用动态运行时配置.清单 1 是对 /proc 中部分元素进行一次交互查询的结果.它显示的是 /proc 文件系统的根目录中的内容.注意,在左边是一系列数字编号的文件.每个实际上都是一个目录,表示系统中的一个进程.由于在 GNU/Linux 中创建的第一个进程是 init 进程,因此它的 process-id 为 1.然后对这个目录执行一个 ls 命令,这会显示很多文件.每个文件都提供了有关这个特殊进程的详细信息.

/proc 中另外一些有趣的文件有：`cpuinfo`，它标识了处理器的类型和速度；`pci`，显示在 PCI 总线上找到的设备；`modules`，标识了当前加载到内核中的模块。

5 实验五 文件系统

5.1 实验目的

设计并实现一个模拟的文件系统.

5.2 实验内容

基于一个大文件(如 100M), 模拟磁盘
格式化, 建立文件系统管理数据结构
实现文件/目录创建/删除, 目录显示等基本功能

5.3 实验设计

5.3.1 实验环境

Python 3.6.6
Prettytable 库
Pickle 库
Logging 库

5.3.2 实验设计

考虑简单的文件系统,仅表现文件信息.
文件结构以`json`序列化实现.
实现从文件序列化读写,并在操作完毕退出环境时写入文件.
提供 CMD 和命令解析.
提供 user 和 root 的用户切换,并且设置权限.

5.3.3 命令列表

系统实现下列命令:
mkfs #格式化磁盘
ls [None]
pwd [None]
cd [foldername]
rm [filename]
su #Switch between root and user

```
mkdir [foldername]
touch [filename]
cat [filename]#Return success or fail
exit#此时再写入文件
```

5.3.4 数据结构

考虑到序列化的要求,采用类 json 的数据结构存取文件数据

```
1. {
2.   "folder":{
3.     "type":"Directory",
4.     "ctime":"timestamp",
5.     "mtime":"timestamp",
6.     "owner":None,
7.     "size":obj,
8.     "subfile1":{},
9.     "subfile2":{}
10.  },
11.  "file":{
12.    "type":"File",
13.    "ctime":"timestamp",
14.    "mtime":"timestamp",
15.    "owner":"user/root",
16.    "size":"fake data"
17.  }
18. }
```

5.4 技术细节

5.4.1 类的继承和调用

为了开发方便并且方便定位错误,这里使用多级类的方式进行处理.类之间有继承关系.详细调用关系如下:

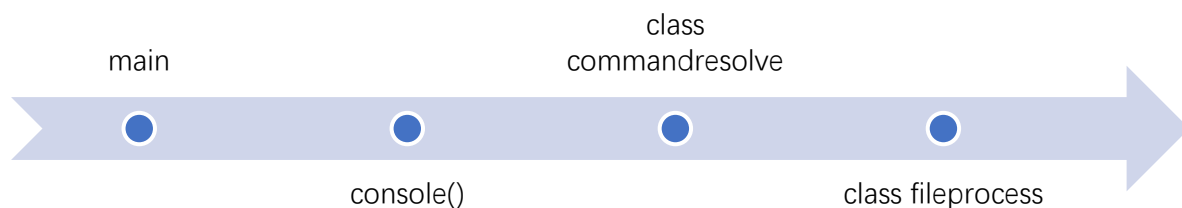


图 5-1

其中类 `commandresolve` 是 `fileprocess` 的子类,在此基础上进行改写和新增功能.

5.4.2 日志记录

使用 `logger` 库记录日志并提供提示信息. 打印日志是很多程序的重要需求, 良好的日志输出可以帮我们更方便的检测程序运行状态. 日志级别有如下几种. 当获取根 `Logger` 的时候, 默认级别为 `NOTSET`, 这样会显示所有输出. 当获取非根 `Logger` 的时候, 根 `Logger` 的默认级别是 `WARNING`, 非根 `Logger` 会继承这个级别, 只有 `WARNING` 以上的日志才会输出.

日志分终端日志 `handle` 和文件日志 `handle`, 分别将日志输出到终端和文件中. 这里使用的是终端日志.

```

1. logger = logging.getLogger()#创建对象
2. logger.setLevel(logging.INFO)#设定起始显示级别
3.
4. # 创建 Handler
5. # 终端 Handler
6. consoleHandler = logging.StreamHandler()
7. consoleHandler.setLevel(logging.INFO)
8. # Formatter
9. formatter = logging.Formatter('%(asctime)s [%(levelname)s] \t %(message)s')
10. consoleHandler.setFormatter(formatter)
11. # 添加到 Logger 中
12. logger.addHandler(consoleHandler)

```

部署时调用 `logger.logger.[level](Message)` 就可以格式化输出调试信息, 并且可以加入任何变量.

比如:

```

2020-06-21 21:54:17,727 [INFO] Data write successfully
2020-06-21 21:54:17,727 [INFO] Exit Successfully

```

图 5-2

5.4.3 命令解析

为了解析方便,这里采用了一种比较基础的解析方法.得到原始输入 `raw` 之后使用 `split` 方法分割为列表.比对第一个表项是否在合法指令列表中,如果在,则根据相关命令规则进行后续表项的检查和相关功能函数的执行.

该部分代码如下:

```
1. def __init__(self,data:dict,logger,):
2.     self.data=data
3.     self.path=data
4.     self.auth=False
5.     self.logger=logger
6.     self.cmd=['mkfs','ls','pwd','cd','rm','su','mkdir','touch','cat','exit']
7.     def resolvecommand(self,rawcommand):
8.         self.raw=rawcommand
9.         cmdlist=self.raw.split()
10.        if cmdlist[0] not in self.cmd:
11.            self.logger.error("Command not found")
12.            return True
13.        name=cmdlist[0]
14.        if name=='mkfs':
15.            self.mkfs()
16.            return True
17.        elif name=='ls':
18.            self.ls()
19.            return True
```

5.4.4 列表的美化输出

为了更清楚的显示命令 `ls` 的内容,这里使用 `prettytable` 库进行美化.

`PrettyTable` 是一个简单的 Python 库,旨在使在具有吸引力的 ASCII 表中快速轻松地表示表格数据。它的灵感来自 PostgreSQL 外壳程序 `psql` 中使用的 ASCII 表。`PrettyTable` 允许选择要打印的列,列的独立对齐(左对齐或右对齐或居中)以及通过指定行范围来打印“子表”

初始化该表需要指定表头,并通过 `add_row` 加入行表项.

```
1. table = PrettyTable(['Name','Type','Create Time','Change Time','Owner','Size'])
2.     for obj in self.path:
3.         if obj in ['name','type','ctime','mtime','owner','size']:
4.             continue
```

```
5.         table.add_row([obj,self.path[obj]['type'],self.path[obj]['ctime'
],self.path[obj]['mtime'],self.path[obj]['owner'],self.path[obj]['size']])
```

输出效果如下:

Name	Type	Create Time	Change Time	Owner	Size
t1	Directory	20-06-21 21:52:48	20-06-21 21:52:48	None	None
p1	File	20-06-21 21:52:51	20-06-21 21:52:51	user	7501058

图 5-3

5.4.5 命名检查

由于目录下目录属性和子文件是在一个层级的,所以如果文件命名和属性名称相同会造成冲突.所以在新建文件或目录时要检查名称有无冲突,包括和保留字冲突以及和现存文件/目录冲突.

相关检查代码如下:

```
1. #Check if the name vaild
2.     if name in ['type','ctime','mtime','owner','size']:
3.         self.logger.error("Name is not permitted")
4.         return
5.     if name in self.path:
6.         self.logger.error("Name already exist")
7.         return
```

5.4.6 数据的序列化和持久化存储

为了方便序列化考虑,这里使用了类 json 格式进行操作.

序列化存储用到的是 pickle 库. 为了将一个对象保存到一个文件中,可以使用 dump 方法;而为了从字节流中恢复一个对象,要使用 pickle.load() 或 pickle.loads() 函数.

对于大多数应用程序来讲, dump() 和 load() 函数的使用就是你有效使用 pickle 模块所需的全部了. 它可适用于绝大部分 Python 数据类型和用户自定义类的对象实例. 如果你碰到某个库可以让你在数据库中保存/恢复 Python 对象或者是通过网络传输对象的话, 那么很有可能这个库的底层就使用了 pickle 模块.

这里使用到该模块的地方:

载入数据:

```
1. if not os.path.isfile(filename):
2.     logger.warning("File not exist. Trying to create...")
3.     createfile(filename,20000000)
4.     with open(filename,'wb') as p:
```

```

5.         pickle.dump({},p)
6.     #Build simlink
7.     data=dict()
8.     with open(filename,"rb") as f:
9.         data=pickle.load(f)

```

装载数据:

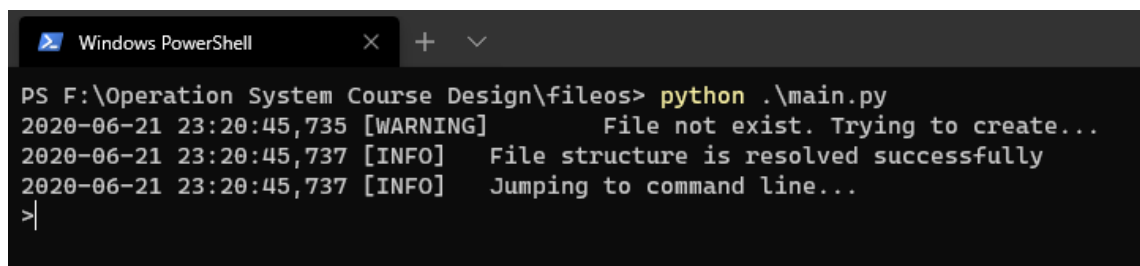
```

1. def exit(self):
2.     with open('simdisk.bin',"wb") as f:
3.         pickle.dump(self.data,f)
4.         self.logger.debug("%s"%self.data)
5.         self.logger.info("Data write successfully")
6.     return

```

5.5 实验测试

从没有模拟文件 simdisk.bin 开始.
初次进入,创建 simdisk 并且准备文件
进入命令行模式.



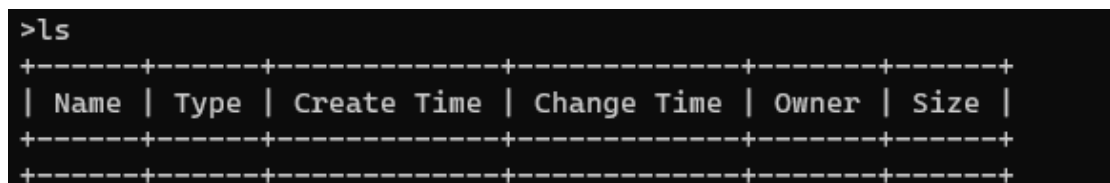
```

Windows PowerShell
PS F:\Operation System Course Design\fileos> python .\main.py
2020-06-21 23:20:45,735 [WARNING]      File not exist. Trying to create...
2020-06-21 23:20:45,737 [INFO]       File structure is resolved successfully
2020-06-21 23:20:45,737 [INFO]       Jumping to command line...
>

```

图 5-4

一开始的文件夹是没有内容的.



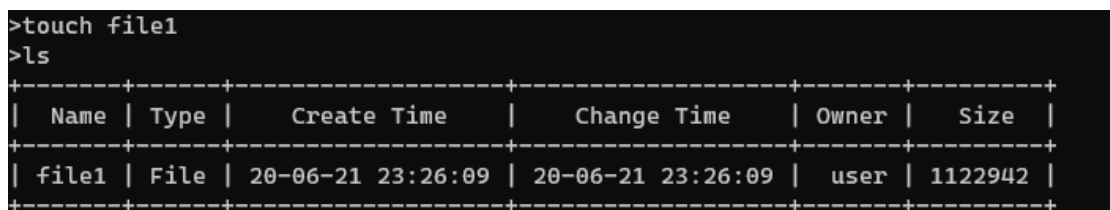
```

>ls
+-----+-----+-----+-----+-----+-----+
| Name | Type | Create Time | Change Time | Owner | Size |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+

```

图 5-5

初次进入系统默认是 user 用户.尝试创建文件.



```

>touch file1
>ls
+-----+-----+-----+-----+-----+-----+
| Name | Type | Create Time | Change Time | Owner | Size |
+-----+-----+-----+-----+-----+-----+
| file1 | File | 20-06-21 23:26:09 | 20-06-21 23:26:09 | user | 1122942 |
+-----+-----+-----+-----+-----+-----+

```

图 5-6

创建成功.

切换为 root 用户,创建文件夹.

```
>su
>mkdir directory
>ls
+-----+-----+-----+-----+-----+-----+
| Name | Type | Create Time | Change Time | Owner | Size |
+-----+-----+-----+-----+-----+-----+
| file1 | File | 20-06-21 23:26:09 | 20-06-21 23:26:09 | user | 1122942 |
| directory | Directory | 20-06-21 23:28:45 | 20-06-21 23:28:45 | None | None |
+-----+-----+-----+-----+-----+-----+
>
```

图 5-7

创建 root 特权文件 file2 并删除 file1.

```
>touch file2
>rm file1
2020-06-21 23:29:48,794 [INFO] file1 has been removed
>ls
+-----+-----+-----+-----+-----+-----+
| Name | Type | Create Time | Change Time | Owner | Size |
+-----+-----+-----+-----+-----+-----+
| directory | Directory | 20-06-21 23:28:45 | 20-06-21 23:28:45 | None | None |
| file2 | File | 20-06-21 23:29:44 | 20-06-21 23:29:44 | root | 5614535 |
+-----+-----+-----+-----+-----+-----+
```

图 5-8

打开 file2.

```
>cat file2
2020-06-21 23:33:59,942 [INFO] Open file successfully
```

图 5-9

切换回 user,尝试打开 file2

```
>su
>cat file2
2020-06-21 23:34:16,995 [ERROR] Operation not permitted
```

图 5-10

权限被禁止.

切换进 directory,新建文件 file3

```
>cd directory
2020-06-21 23:35:00,543 [INFO] Change Successfully.
>ls
+-----+-----+-----+-----+-----+-----+
| Name | Type | Create Time | Change Time | Owner | Size |
+-----+-----+-----+-----+-----+-----+
>touch file3
>ls
+-----+-----+-----+-----+-----+-----+
| Name | Type | Create Time | Change Time | Owner | Size |
+-----+-----+-----+-----+-----+-----+
| file3 | File | 20-06-21 23:35:06 | 20-06-21 23:35:06 | user | 5443686 |
+-----+-----+-----+-----+-----+-----+
```

图 5-11

目录改变并创建成功.

退出,将修改写入文件.

```
>exit
2020-06-21 23:35:44,511 [INFO] Data write successfully
2020-06-21 23:35:44,512 [INFO] Exit Successfully
```

图 5-12

再次打开,序列化持久化存储成功.

```
PS F:\Operation System Course Design\fileos> python .\main.py
2020-06-21 23:36:17,104 [INFO] Get existed file data, trying to resolve...
2020-06-21 23:36:17,104 [INFO] File structure is resolved successfully
2020-06-21 23:36:17,105 [INFO] Jumping to command line...
>ls
+-----+-----+-----+-----+-----+-----+
| Name | Type | Create Time | Change Time | Owner | Size |
+-----+-----+-----+-----+-----+-----+
| directory | Directory | 20-06-21 23:28:45 | 20-06-21 23:28:45 | None | None |
| file2 | File | 20-06-21 23:29:44 | 20-06-21 23:29:44 | root | 5614535 |
+-----+-----+-----+-----+-----+-----+
```

图 5-13

6 实验总结

这次实验时间跨度比较长,为了保持技术细节的完整,相关的技术文档变得十分必要.在实验的过程中,撰写了大量的技术文档,同时也在探索和改进技术文档的撰写方式.最后除了代码和报告,还积累了大量的技术文档,不管是留作自己的学习资料还是作为他人学习参考都是非常有价值的.

本项目已经在 GitHub 上开源.

https://github.com/IRIDIUM-SUB/Sys_Course_Design

本次实验不可避免的遇到很多问题,尤其是涉及到系统底层的问题.操作系统的内核是一个非常复杂的结构,稍有错误就会导致系统崩溃.本次实验中我学会了查看系统底层的日志,也对操作系统的底层架构有了新的认识.

7 Appendix 实验源码

7.1 实验一

7.1.1 Copy Trigger

Autorun.sh

```
1. g++ -o copytrigger -g copytrigger.cpp
2. echo compile finished
3. ./copytrigger copy src dst
```

Copy Trigger.cpp

```
1. /*
2.  * @Author: I-Hisen
3.  * @Date: 2020-02-15 20:33:04
4.  * @LastEditTime: 2020-02-15 20:36:41
5.  * @LastEditors: Please set LastEditors
6.  * @Description: 编写一个 C 程序，用 read、write 等系统调用实现文件拷贝功能。
7.  * 命令形式: copy <源文件名> <目标文件名>
8.  * ref:https://www.cnblogs.com/acerkoo/p/9490308.html
9.  * ref:https://www.cnblogs.com/smallredness/p/10259237.html
10. * ref:https://www.cnblogs.com/smallredness/p/10259232.html
11. * ref:https://www.purebasic.com/documentation/filesystem/getfileattributes.
    html
12. * ref:https://blog.csdn.net/cs_zlg/article/details/8271741
13. * ref:https://blog.csdn.net/aitcax/article/details/50911445
14. */
15. #include <iostream>
16. #include <cstring>
17. #include <fstream>
18. #include <unistd.h>
19. #include <sys/stat.h>
20. using namespace std;
21. bool fileexist(const char * path);
22. int filesize(const char *fname);
23. int main(int argc, char *argv[])
24. {
25.
26.     if (argc != 4)
27.     {
```



```

69.         } //to locate eof
70.     }
71.     outfile.write(buf, len); //output
72. }
73. infile.close(); //close
74. outfile.flush(); //flush stream
75. clog<<"outfile flushed"<<endl;
76. outfile.close(); //close
77. clog << "Process Finished" << endl;
78. return 0;
79. }
80. else if (pad == "U" || pad == "u")
81. {
82.     //Using C style to process
83.     //But... text or binary?
84. bd:
85.     cout << "Binary or Document?\n[B/D]" << endl;
86.     string bd;
87.     cin >> bd;
88.     if (bd == "B" || bd == "b")
89.     {
90.         cout << "Comfirm:\n\tBinary Mode" << endl;
91.         FILE *f1 = fopen(src.c_str(), "rb");
92.         FILE *f2 = fopen(dst.c_str(), "wb");
93.         if (!f1 || !f2) //if open successfully
94.         {
95.             cerr << "Error: Fail to open file" << endl;
96.             return 0;
97.         }
98.         char buf[1024];
99.         int len = sizeof(buf);
100.        while (!feof(f1))
101.        {
102.            memset(buf, 0, sizeof(buf)); //initialize buffer
103.            if (fread(buf, sizeof(buf), 1, f1) != 1)
104.            {
105.                char *p = &buf[len - 1];
106.                while ((*p) == 0)
107.                {
108.                    p--;
109.                    len--;
110.                }
111.            }
112.            fwrite(buf, len, 1, f2);

```

```

113.         }
114.         fclose(f1);
115.         fclose(f2);
116.         clog << "Binary Mode Succeed" << endl;
117.         return 0;
118.     }
119.     else if (bd == "D" || bd == "d")
120.     {
121.         cout << "Comfirm:\n\tDocument Mode" << endl;
122.         FILE *f1 = fopen(src.c_str(), "r");
123.         FILE *f2 = fopen(dst.c_str(), "w");
124.         if (!f1 || !f2)
125.         {
126.             cerr << "Error: Fail to open file" << endl;
127.             return -1;
128.         }
129.         char buf[50];
130.         while (fgets(buf, sizeof(buf), f1) != NULL)
131.         {
132.             cout<<buf<<endl;
133.             fputs(buf, f2);
134.         }
135.         fclose(f1);
136.         fclose(f2);
137.         clog << "Docunent Mode Succeed" << endl;
138.     }
139.     else
140.     {
141.         goto bd;
142.     }
143. }
144. else if(pad=="n" || pad=="N")
145. {
146.     clog<<"Quit without Operation"<<endl;
147.     return 0;
148. }
149. else
150. {
151.     goto choosemod;
152. }
153. }
154. bool fileexist(const char *path)
155. {
156.     if (access(path, 0)==0)//0=success -1=failed

```

```

157.     {
158.         return true;
159.     }
160.     else
161.     {
162.         return false;
163.     }
164. }
165. int filesize(const char *fname)
166. {
167.     struct stat statbuf;
168.     if (stat(fname, &statbuf) == 0)
169.         return statbuf.st_size;
170.     else
171.         return -1;
172. }

```

7.1.2 Multiprocessing Demo

Main.cpp

```

1. #include "widget.h"
2. #include "widget1.h"
3. #include "widget2.h"
4. #include <QApplication>
5. #include <stdio.h>
6. #include <stdlib.h>
7. #include <unistd.h>
8.
9. int main(int argc, char *argv[])
10. {
11.     int pid;
12.     if((pid = fork()) == 0){
13.         QApplication a(argc, argv); //Create an application
14.         widget1 w; //Create a widget
15.         w.move(100,200);
16.         w.setWindowTitle("Counter");
17.         w.show();
18.         a.exec();
19.         exit(0);
20.     }
21.     if((pid = fork()) == 0){

```

```

22.     QApplication a(argc, argv);
23.     widget2 w;
24.     w.move(300,400);
25.     w.setWindowTitle("Accumulator");
26.     w.show();
27.     a.exec();
28.     exit(0);
29. }
30. QApplication a(argc, argv);
31. Widget w;
32. w.move(500,600);
33. w.setWindowTitle("Clock");
34. w.show();
35. return a.exec();
36. }

```

Widget.h

```

1. #ifndef WIDGET_H
2. #define WIDGET_H
3.
4. #include <QWidget>
5.
6. namespace Ui {
7. class Widget;
8. }
9.
10. class Widget : public QWidget
11. {
12.     Q_OBJECT
13.
14. public:
15.     explicit Widget(QWidget *parent = nullptr);
16.     ~Widget();
17.
18. public slots:
19.     void Update();
20.
21. private:
22.     Ui::Widget *ui;
23. };
24.
25. #endif // WIDGET_H

```

Wiget1.h

```
1. #ifndef WIDGET1_H
2. #define WIDGET1_H
3.
4. #include <QWidget>
5.
6. namespace Ui {
7. class widget1;
8. }
9.
10. class widget1 : public QWidget
11. {
12.     Q_OBJECT
13.
14. public:
15.     explicit widget1(QWidget *parent = nullptr);
16.     ~widget1();
17.
18. public slots:
19.     void Update();
20.
21. private:
22.     int num;
23.     Ui::widget1 *ui;
24. };
25.
26. #endif // WIDGET1_H
```

Wiget2.h

```
1. #ifndef WIDGET2_H
2. #define WIDGET2_H
3.
4. #include <QWidget>
5.
6. namespace Ui {
7. class widget2;
8. }
9.
10. class widget2 : public QWidget
11. {
12.     Q_OBJECT
13.
14. public:
```



```

15.     explicit widget2(QWidget *parent = nullptr);
16.     ~widget2();
17.
18. public slots:
19.     void Update();
20.
21. private:
22.     int sum, num;
23.     Ui::widget2 *ui;
24. };
25.
26. #endif // WIDGET2_H

```

Widget.ui

```

1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <ui version="4.0">
3.    <class>Widget</class>
4.    <widget class="QWidget" name="Widget">
5.      <property name="geometry">
6.        <rect>
7.          <x>0</x>
8.          <y>0</y>
9.          <width>505</width>
10.         <height>290</height>
11.        </rect>
12.      </property>
13.      <property name="windowTitle">
14.        <string>Widget</string>
15.      </property>
16.      <widget class="QLabel" name="label_5">
17.        <property name="geometry">
18.          <rect>
19.            <x>50</x>
20.            <y>40</y>
21.            <width>144</width>
22.            <height>32</height>
23.          </rect>
24.        </property>
25.        <property name="font">
26.          <font>
27.            <pointsize>20</pointsize>
28.            <underline>true</underline>
29.          </font>
30.        </property>

```

```
31. <property name="text">
32.   <string>Current PID</string>
33. </property>
34. <property name="alignment">
35.   <set>Qt::AlignCenter</set>
36. </property>
37. </widget>
38. <widget class="QLabel" name="label_6">
39.   <property name="geometry">
40.     <rect>
41.       <x>20</x>
42.       <y>200</y>
43.       <width>211</width>
44.       <height>41</height>
45.     </rect>
46.   </property>
47.   <property name="font">
48.     <font>
49.       <pointsize>20</pointsize>
50.       <underline>true</underline>
51.     </font>
52.   </property>
53.   <property name="text">
54.     <string>Current Time</string>
55.   </property>
56.   <property name="alignment">
57.     <set>Qt::AlignCenter</set>
58.   </property>
59. </widget>
60. <widget class="QLCDNumber" name="lcdNumber_2">
61.   <property name="geometry">
62.     <rect>
63.       <x>320</x>
64.       <y>40</y>
65.       <width>121</width>
66.       <height>41</height>
67.     </rect>
68.   </property>
69. </widget>
70. <widget class="QLCDNumber" name="lcdNumber_3">
71.   <property name="geometry">
72.     <rect>
73.       <x>280</x>
74.       <y>200</y>
```

```

75.     <width>201</width>
76.     <height>41</height>
77.     </rect>
78. </property>
79. <property name="digitCount">
80.     <number>8</number>
81. </property>
82. </widget>
83. <widget class="Line" name="line">
84.     <property name="geometry">
85.         <rect>
86.             <x>47</x>
87.             <y>130</y>
88.             <width>431</width>
89.             <height>20</height>
90.         </rect>
91.     </property>
92.     <property name="orientation">
93.         <enum>Qt::Horizontal</enum>
94.     </property>
95. </widget>
96. </widget>
97. <layoutdefault spacing="6" margin="11"/>
98. <resources/>
99. <connections/>
100. </ui>

```

Widget1.ui

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <ui version="4.0">
3.     <class>widget1</class>
4.     <widget class="QWidget" name="widget1">
5.         <property name="geometry">
6.             <rect>
7.                 <x>0</x>
8.                 <y>0</y>
9.                 <width>505</width>
10.                <height>290</height>
11.            </rect>
12.        </property>
13.        <property name="windowTitle">
14.            <string>Widget</string>
15.        </property>
16.        <widget class="QLCDNumber" name="lcdNumber_4">

```

```
17. <property name="geometry">
18. <rect>
19. <x>310</x>
20. <y>30</y>
21. <width>121</width>
22. <height>41</height>
23. </rect>
24. </property>
25. </widget>
26. <widget class="QLabel" name="label_5">
27. <property name="geometry">
28. <rect>
29. <x>40</x>
30. <y>30</y>
31. <width>144</width>
32. <height>32</height>
33. </rect>
34. </property>
35. <property name="font">
36. <font>
37. <pointsize>20</pointsize>
38. <underline>true</underline>
39. </font>
40. </property>
41. <property name="text">
42. <string>Current PID</string>
43. </property>
44. <property name="alignment">
45. <set>Qt::AlignCenter</set>
46. </property>
47. </widget>
48. <widget class="Line" name="line">
49. <property name="geometry">
50. <rect>
51. <x>37</x>
52. <y>120</y>
53. <width>431</width>
54. <height>20</height>
55. </rect>
56. </property>
57. <property name="orientation">
58. <enum>Qt::Horizontal</enum>
59. </property>
60. </widget>
```

```

61. <widget class="QLCDNumber" name="lcdNumber_5">
62.   <property name="geometry">
63.     <rect>
64.       <x>350</x>
65.       <y>190</y>
66.       <width>41</width>
67.       <height>41</height>
68.     </rect>
69.   </property>
70.   <property name="digitCount">
71.     <number>1</number>
72.   </property>
73. </widget>
74. <widget class="QLabel" name="label_6">
75.   <property name="geometry">
76.     <rect>
77.       <x>10</x>
78.       <y>190</y>
79.       <width>211</width>
80.       <height>41</height>
81.     </rect>
82.   </property>
83.   <property name="font">
84.     <font>
85.       <pointsize>20</pointsize>
86.       <underline>true</underline>
87.     </font>
88.   </property>
89.   <property name="text">
90.     <string>Current Counter</string>
91.   </property>
92.   <property name="alignment">
93.     <set>Qt::AlignCenter</set>
94.   </property>
95. </widget>
96. </widget>
97. <layoutdefault spacing="6" margin="11"/>
98. <resources/>
99. <connections/>
100. </ui>

```

Widget2.ui

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <ui version="4.0">

```

```
3.  <class>widget2</class>
4.  <widget class="QWidget" name="widget2">
5.    <property name="geometry">
6.      <rect>
7.        <x>0</x>
8.        <y>0</y>
9.        <width>560</width>
10.       <height>349</height>
11.      </rect>
12.    </property>
13.    <property name="windowTitle">
14.      <string>Widget</string>
15.    </property>
16.    <widget class="QLCDNumber" name="lcdNumber_7">
17.      <property name="geometry">
18.        <rect>
19.          <x>340</x>
20.          <y>110</y>
21.          <width>121</width>
22.          <height>41</height>
23.        </rect>
24.      </property>
25.      <property name="digitCount">
26.        <number>4</number>
27.      </property>
28.    </widget>
29.    <widget class="QLabel" name="label_7">
30.      <property name="geometry">
31.        <rect>
32.          <x>20</x>
33.          <y>110</y>
34.          <width>211</width>
35.          <height>41</height>
36.        </rect>
37.      </property>
38.      <property name="font">
39.        <font>
40.          <pointsize>20</pointsize>
41.          <underline>true</underline>
42.        </font>
43.      </property>
44.      <property name="text">
45.        <string>Current Num</string>
46.      </property>
```

```
47.     <property name="alignment">
48.         <set>Qt::AlignCenter</set>
49.     </property>
50. </widget>
51. <widget class="QLabel" name="label_8">
52.     <property name="geometry">
53.         <rect>
54.             <x>53</x>
55.             <y>30</y>
56.             <width>144</width>
57.             <height>32</height>
58.         </rect>
59.     </property>
60.     <property name="font">
61.         <font>
62.             <pointsize>20</pointsize>
63.             <underline>true</underline>
64.         </font>
65.     </property>
66.     <property name="text">
67.         <string>Current PID</string>
68.     </property>
69.     <property name="alignment">
70.         <set>Qt::AlignCenter</set>
71.     </property>
72. </widget>
73. <widget class="QLCDNumber" name="lcdNumber_6">
74.     <property name="geometry">
75.         <rect>
76.             <x>340</x>
77.             <y>30</y>
78.             <width>121</width>
79.             <height>41</height>
80.         </rect>
81.     </property>
82. </widget>
83. <widget class="Line" name="line">
84.     <property name="geometry">
85.         <rect>
86.             <x>10</x>
87.             <y>190</y>
88.             <width>521</width>
89.             <height>20</height>
90.         </rect>
```

```
91.     </property>
92.     <property name="orientation">
93.         <enum>Qt::Horizontal</enum>
94.     </property>
95. </widget>
96. <widget class="QLCDNumber" name="lcdNumber_8">
97.     <property name="geometry">
98.         <rect>
99.             <x>240</x>
100.            <y>240</y>
101.            <width>301</width>
102.            <height>71</height>
103.        </rect>
104.    </property>
105.    <property name="digitCount">
106.        <number>6</number>
107.    </property>
108. </widget>
109. <widget class="QLabel" name="label_9">
110.     <property name="geometry">
111.         <rect>
112.             <x>30</x>
113.             <y>241</y>
114.             <width>161</width>
115.             <height>51</height>
116.         </rect>
117.     </property>
118.     <property name="font">
119.         <font>
120.             <pointsize>20</pointsize>
121.             <underline>true</underline>
122.         </font>
123.     </property>
124.     <property name="text">
125.         <string>Sum</string>
126.     </property>
127.     <property name="alignment">
128.         <set>Qt::AlignCenter</set>
129.     </property>
130. </widget>
131. </widget>
132. <layoutdefault spacing="6" margin="11"/>
133. <resources/>
134. <connections/>
```


135. </ui>

Widget.cpp

```
1. #include "widget.h"
2. #include "ui_widget.h"
3. #include <unistd.h>
4. #include <QDateTime>
5. #include <QTimer>
6.
7. Widget::Widget(QWidget *parent) :
8.     QWidget(parent),
9.     ui(new Ui::Widget)
10. {
11.     ui->setupUi(this);
12.     this->setFixedSize(this->width(), this->height()); //Unable Adjustment of
        size
13.     this->move(600, 400);
14.
15.     int pid = getpid();
16.     //ui->label_2->setText(QString::number(pid, 10));
17.     ui->lcdNumber_2->display(pid);
18.
19.     QDateTime curtime = QDateTime::currentDateTime();
20.     //ui->label_4->setText(curtime.toString("hh:mm:ss"));
21.
22.     ui->lcdNumber_3->display(curtime.toString("hh:mm:ss"));
23.     QTimer *timer = new QTimer(this);
24.     connect(timer, SIGNAL(timeout()), this, SLOT(Update()));
25.     timer->start(1000);
26. }
27.
28. void Widget::Update(){
29.     QDateTime curtime = QDateTime::currentDateTime();
30.     ui->lcdNumber_3->display(curtime.toString("hh:mm:ss"));
31.     // ui->label_4->setText(curtime.toString("hh:mm:ss"));
32. }
33.
34. Widget::~Widget()
35. {
36.     delete ui;
37. }
```

Widget1.cpp

```

1. #include "widget1.h"
2. #include "ui_widget1.h"
3. #include <unistd.h>
4. #include <QTimer>
5.
6. widget1::widget1(QWidget *parent) :
7.     QWidget(parent),
8.     ui(new Ui::widget1)
9. {
10.     ui->setupUi(this);
11.     this->setFixedSize(this->width(), this->height());
12.     this->move(900,400);
13.
14.     int pid = getpid();
15.     //ui->label_2->setText(QString::number(pid, 10));
16.     ui->lcdNumber_4->display(pid);
17.     num = 0;
18.     //ui->label_4->setText(QString::number(num, 10));
19.     ui->lcdNumber_5->display(num);
20.     QTimer *timer = new QTimer(this);
21.     connect(timer, SIGNAL(timeout()), this, SLOT(Update()));
22.     timer->start(1000);
23. }
24.
25. void widget1::Update(){
26.     num = (num +1) % 10;
27.     ui->lcdNumber_5->display(num);
28.     //ui->label_4->setText(QString::number(num, 10));
29. }
30.
31. widget1::~~widget1()
32. {
33.     delete ui;
34. }

```

Widget2.cpp

```

1. #include "widget2.h"
2. #include "ui_widget2.h"
3. #include <unistd.h>
4. #include <QTimer>
5.
6. widget2::widget2(QWidget *parent) :
7.     QWidget(parent),
8.     ui(new Ui::widget2)

```

```

9. {
10.     ui->setupUi(this);
11.     this->setFixedSize(this->width(), this->height());
12.     this->move(1200, 400);
13.
14.     int pid = getpid();
15.     ui->lcdNumber_6->display(pid);
16.
17.     sum = 0;
18.     num = 1;
19.     ui->lcdNumber_8->display(sum);
20.
21.     QTimer *timer = new QTimer(this);
22.     connect(timer, SIGNAL(timeout()), this, SLOT(Update()));
23.     timer->start(1000);
24. }
25.
26. void widget2::Update(){
27.     if(num <= 1000)
28.         sum += num++;
29.     ui->lcdNumber_8->display(sum);
30.     ui->lcdNumber_7->display(num-1);
31. }
32. widget2::~~widget2()
33. {
34.     delete ui;
35. }

```

7.2 实验二

7.2.1 业务代码部分

Sys.c

```

1. asmlinkage int sys_customcp(const char *src, const char *dst)
2. {
3.     int infd, outfd, count;
4.     char buf[256];
5.     mm_segment_t fs;
6.     fs = get_fs();
7.     set_fs(get_ds( ));
8.     //Handle Error
9.     if((infd=sys_open(src, O_RDONLY, 0)) == -1)

```

```

10.  {
11.      return 1;
12.  }
13.  if((outfd=sys_open(dst, O_WRONLY | O_CREAT, S_IRUSR| S_IWUSR)) ==-1)
14.  {
15.      return 2;
16.  }
17.  while((count = sys_read(infd, buf, 256)) > 0)
18.  {
19.      if(sys_write(outfd, buf, count) != count)
20.          return 3;
21.  }
22.  if(count == -1)
23.      return 4;
24.  sys_close(infd);
25.  sys_close(outfd);
26.  set_fs(fs);
27.  return 0;
28. }

```

Syscall.h

```

1.  asmlinkage int sys_customcp(const char *src,const char *dst);

```

7.2.2 部署部分

Autodeploy.sh

```

1.  # Env Check
2.  sudo apt-get update
3.  sudo apt-get install gcc
4.  sudo apt-get install make
5.  sudo apt-get install build-essential
6.  sudo apt-get install ncurses-dev
7.  sudo apt-get install libssl-dev
8.  echo "Environment Check Finished..."
9.
10. # Deploy
11. sudo cp -fp syscall_64.tbl ../linux-4.15.2/arch/x86/entry/syscalls/
12. sudo cp -fp syscalls.h ../linux-4.15.2/include/linux/
13. sudo cp -fp sys.c ../linux-4.15.2/kernel/
14. echo "Finished"

```

Autocompile.sh

```
1. cd ..
2. cd linux-4.15.2
3. echo "Start"
4. sudo make clean
5. sudo make -j4 bzImage
6. echo "Image Compile Finished"
7. sudo make -j4 modules
8. echo "Modules Finished"
9. sudo make -j4 modules_install
10. echo "Module Installed"
11. sudo make -j4 install
12. echo "Finished"
```

7.2.3 利用部分

Shellcode.cpp

```
1. #include <stdio.h>
2. #include <linux/kernel.h>
3. #include <unistd.h>
4. #include <string.h>
5. int main(int argc, char* argv[])
6. {
7.     int status=114514;
8.     if (argc!=4||!strcmp(argv[0],"copy"))
9.     {
10.         printf("Usage: copy <src> <dst>\n");
11.         return 0;
12.     }
13.     else
14.     {
15.         printf("Confirm:%s %s %s\n",argv[1],argv[2],argv[3]);
16.         status=syscall(333,argv[2],argv[3]);
17.         if (status==1)
18.         {
19.             printf("Failed: Unable to open arc file\n");
20.         }
21.         else if (status==2)
22.         {
23.             printf("Failed: Unable to open dst file\n");
24.         }
25.         else if (status==3)
26.         {
27.             printf("Failed: Buffer Overflow\n");
```

```

28.     }
29.     else if (status==4)
30.     {
31.         printf("Failed: Error while cpying\n");
32.     }
33.     else if (!status)
34.     {
35.         printf("Success!\n");
36.     }
37.     return 0;
38. }
39. printf("%d\n",status);
40. return 0;
41. }

```

7.3 实验三

Basindev.c

```

1. #include <linux/init.h>
2. #include <linux/module.h>
3. #include <linux/types.h>
4. #include <linux/fs.h>
5. #include <linux/sched.h>
6. #include <linux/cdev.h>
7. #include <linux/slab.h>
8. #include <linux/uaccess.h> //copy_to_user
9.
10. #define DEV_SIZE 2048
11. MODULE_LICENSE("GPL");
12. static int dev_major_number = 0;
13.
14. unsigned char mybuf[DEV_SIZE] = "customdev by POTASSIUM";
15.
16. static ssize_t my_read(struct file* fp, char __user* userbuf, size_t count, lo
    ff_t* pos){
17.     int size = count;
18.     if(size>DEV_SIZE){
19.         printk(KERN_ALERT "out of max");
20.         size = DEV_SIZE;
21.     }
22.     if(copy_to_user(userbuf,mybuf,size)){
23.         return -ENOMEM;
24.     }

```

```

25.     return size;
26. }
27.
28. static ssize_t my_write(struct file* fp, const char __user* userbuf, size_t co
    unt, loff_t* pos){
29.     int size = count;
30.     if(size>DEV_SIZE){
31.         printk(KERN_ALERT "out of max");
32.         size = DEV_SIZE;
33.     }
34.     if(copy_from_user(mybuf, userbuf, size)){
35.         return -ENOMEM;
36.     }
37.     return size;
38. }
39. static struct file_operations customdev_fops =
40. {
41.     //.owner    = THIS_MODULE,
42.     .read      = my_read,
43.     .write     = my_write,
44. };
45. static int my_init(void)
46. {
47.     int result = 0;
48.     printk( KERN_NOTICE "register_device() is called." );
49.     result = register_chrdev(0, "customdev", &customdev_fops);
50.
51.     if(result<0)
52.     {
53.         printk("customdev:can\'t register character device with errorcod
            e = %i", result);
54.         return result;
55.     }
56.
57.     dev_major_number = result;
58.     printk( KERN_NOTICE "customdev: Major = %i \n", dev_major_number);
59.     return 0;
60.
61. }
62.
63. static void my_exit(void)
64. {
65.     unregister_chrdev(dev_major_number, "customdev");
66.     printk("<1>" "unregister customdev succeed!\n");

```

```
67.     return;
68. }
69.
70. module_init(my_init);
71. module_exit(my_exit);
```

Utilize.c

```
1. #include <sys/types.h>
2. #include <sys/stat.h>
3. #include <stdlib.h>
4. #include <string.h>
5. #include <stdio.h>
6. #include <fcntl.h>
7. #include <unistd.h>
8. #define MAX_SIZE 1024
9.
10. int main(void)
11. {
12.     int customdev;
13.     char buf[MAX_SIZE];    //缓冲区
14.     char get[MAX_SIZE];    //要写入的信息
15.
16.     char dir[50] = "/dev/customdev";    //设备名
17.
18.     customdev = open(dir, O_RDWR | O_NONBLOCK);
19.     if (customdev==-1)
20.     {
21.         printf("Cann't open file \n"); exit(0);
22.         return -1;
23.     }
24.     //读初始信息
25.     read(customdev, buf, sizeof(buf));
26.     printf("%s\n", buf);
27.
28.     //写信息
29.     printf("input :");
30.     gets(get);
31.     write(customdev, get, sizeof(get));
32.
33.     //读刚才写的信息
34.     read(customdev, buf, sizeof(buf));
35.     printf("output: %s\n", buf);
36.
37.     close(customdev);
```



```
38.     return 0;
39.
40. }
```

Makefile

```
1. ifneq ($(KERNELRELEASE),)
2. #kbuild syntax.
3. mymodule-objs :=basindev.o
4. obj-m :=basindev.o
5. else
6.
7. PWD :=$(shell pwd)
8. KVER :=$(shell uname -r)
9. KDIR :=/lib/modules/$(KVER)/build
10. all:
11.     $(MAKE) -C $(KDIR) M=$(PWD)
12. clean:
13.     rm -f *.cmd *.o *.mod *.ko
14. endif
```

autodeploy.sh

```
1. echo "Start deploying";
2. sudo make;
3. sudo insmod basindev.ko;
4. echo "Check dev:";
5. sudo mknod /dev/customdev c 240 0;
6. sudo cat /proc/devices| grep "customdev";
7. echo "Deploy finished!";
```

uninstall.sh

```
1. sudo make clean;
2. sudo rmmod basindev;
3. sudo rm -r /dev/customdev;
4. echo "Uninstall Successfully";
```

7.4 实验四

Main.py

```
1. from tkinter import *
2. from tk import *
```

```

3.  """
4.  了解/proc 文件的特点和使用方法
5.  监控系统状态，显示系统部件的使用情况
6.  用图形界面监控系统状态，包括系统基本信息、CPU 和内存利用率、所有进程信息等(可自己补充、添加其他功能)
7.  """
8.
9.
10. if __name__=="__main__":
11.     """
12.     main entrance
13.     """
14.     tkobj=tk()
15.     tkobj.root.mainloop()

```

info.py

```

1. from os import *
2. from re import *
3. from pprint import *# Unit test
4. def analyze(raw:list,itemlist:list):
5.     """
6.     Sort and find key words and return a aorted list
7.     """
8.     itemlen=len(itemlist)
9.     res=["" for _ in range(itemlen)]
10.    final=dict()
11.    for item in raw:
12.        if itemlist==["" for _ in range(itemlen)]:
13.            break#Finish
14.        else:
15.            for opt in itemlist:
16.                if opt=="":
17.                    continue
18.                if item.find(opt)!=-1:#Match
19.                    res[itemlist.index(opt)]=item
20.                    itemlist[itemlist.index(opt)]=""
21.                    break
22.
23.    for i in range(len(res)):
24.        res[i]=sub('\s','',res[i]) #Sort
25.    for item in res:#TODO: Revise to reduce circulations
26.        temp=item.split(":")
27.        final[temp[0]]=temp[1]
28.    return final

```

```
29. def getpid():
30.     ....
31.     Return pidinfo
32.     ...
33.     exec=popen("ls /proc -F|grep [0-9]*")
34.     raw=exec.readlines()
35.     res=findall("[0-9]+",str(raw))
36.
37.     pid=list()
38.     for item in res:
39.         pid.append(int(item))
40.     return set(pid)#Deduplication
41. def getpidinfo(pid:int):
42.     command="cat /proc/"+str(pid)+"/status"
43.     exec=popen(command)
44.     raw=exec.readlines()
45.
46.     if raw ==[]:#Exception
47.         return False
48.     itemlist=["Name", "PPid", "State", "Threads"]
49.     return analyze(raw,itemlist)
50.
51. def getcpuinfo():
52.     command="cat /proc/cpuinfo"
53.     exec=popen(command)
54.     raw=exec.readlines()
55.
56.     itemlist=["vendo", "family", "model", "MHz", "cpu cores"]
57.     return analyze(raw,itemlist)
58.
59. def getmeminfo():
60.     command="cat /proc/meminfo"
61.     exec=popen(command)
62.     raw=exec.readlines()
63.
64.     itemlist=["MemTotal", "MemFree"]
65.     return analyze(raw,itemlist)
66.
67. def getsysinfo():
68.     command="cat /proc/version"
69.     exec=popen(command)
70.     raw=exec.read()
71.     return raw
72.
```

```

73. def getinfo():
74.     '''
75.     Use proc to get information and return dict like:
76.     {
77.         pidinfo:{PID:[Name,Parent PID,State,Threads]}
78.         cpuinfo:{vendor,family,model,frequency,cores},
79.         meminfo:[total,free],
80.         sysinfo:str
81.     }
82.     '''
83.     '''
84.     PID info
85.     '''
86.     info=dict()
87.     pidlist=set(getpid())
88.     result=dict()
89.     for pid in pidlist:
90.         temp=getpidinfo(pid)
91.         if temp==False:#Exception
92.             continue
93.         else:
94.             result[str(pid)]=temp
95.     info['pidinfo']=result
96.     del(result)# Release
97.     '''
98.     CPU info
99.     '''
100.    info['cpuinfo']=getcpuinfo()
101.    '''
102.    Memory info
103.    '''
104.    info['meminfo']=getmeminfo()
105.    '''
106.    System info
107.    '''
108.    info['sysinfo']=getsysinfo()
109.
110.    return info
111.
112.
113. if __name__=="__main__":#Unit test only
114.    pprint(getinfo())

```

tk.py

```

1. from tkinter import *
2. from info import *
3. from multibox import *
4. class tk(object):
5.     def __init__(self):
6.         ....
7.         Main loop of windows
8.         ...
9.         self.root= Tk()
10.        self.buildframe()
11.        self.loadtext()
12.        self.root.title("System monitor")
13.        self.root.geometry("1000x600+10+10")
14.        self.procinfo=""
15.
16.    def buildframe(self):
17.
18.        ....
19.        Root frame
20.        ...
21.        self.frame = Frame(self.root)
22.        ....
23.        Second Layer
24.        ...
25.        self.leftframe=Frame(self.frame,relief="sunken",width=200, height=60
        0)
26.        self.rightframe=Frame(self.frame,relief="sunken",width=600, height=6
        00)
27.        self.botframe=Frame(self.frame,relief="sunken")
28.
29.        ....
30.        Third Layer
31.        ...
32.        self.ltf=Frame(self.leftframe,width=200, height=400)
33.        self.lbf=Frame(self.leftframe,width=200, height=200)
34.
35.
36.        ....
37.        Packing
38.        ...
39.
40.
41.        self.ltf.pack(side=TOP,expand=True,fill=BOTH)
42.        self.lbf.pack(side=BOTTOM,expand=True,fill=BOTH)

```

```

43.
44.         self.botframe.pack(side=BOTTOM,expand=True,fill=BOTH)
45.         self.leftframe.pack(side=LEFT,expand=True,fill=BOTH)
46.         self.rightframe.pack(side=RIGHT,expand=True,fill=BOTH)
47.
48.
49.         self.frame.pack(expand=True,fill=BOTH)
50.
51.
52.
53.     def loadtext(self):
54.         ...
55.         Padding
56.         ...
57.         self.resolvedata()
58.         print("Load Text")
59.         if self.procinfo=="":#Failed, display default text
60.             Label(self.ltf,text='CPUinfo',font=("Arial", 12), borderwidth=2)
        .pack()
61.             Label(self.lbf,text="Meminfo",font=("Arial", 12), borderwidth=2)
        .pack()
62.             Label(self.rightframe,text="PIDinfo",font=("Arial", 12), borderw
        idth=2).pack()
63.             Label(self.botframe,text="Sysinfo",font=("Arial", 12), borderwid
        th=2).pack()
64.
65.         else:
66.             self.cpubox=MultiListbox(self.ltf,self.cpuinfo[0])
67.             for item in self.cpuinfo[1:]:
68.                 self.cpubox.insert(END,item)
69.             self.cpubox.pack(expand=YES, fill=BOTH)
70.
71.             self.pidbox=MultiListbox(self.rightframe,self.pidinfo[0])
72.             for item in self.pidinfo[1:]:
73.                 self.pidbox.insert(END,item)
74.             self.pidbox.pack(expand=YES, fill=BOTH)
75.
76.             self.membox=MultiListbox(self.lbf,self.meminfo[0])
77.             for item in self.meminfo[1:]:
78.                 self.membox.insert(END,item)
79.             self.membox.pack(expand=YES, fill=BOTH)
80.
81.             Label(self.botframe,text=self.sysinfo,font=("Arial", 12), border
        width=2).pack()

```

```

82.
83.
84.     def resolvedata(self):
85.         """
86.         Get info via proc
87.         """
88.         self.procinfo=getinfo()
89.         if type(self.procinfo) is not dict:
90.             return None
91.         """
92.         Sorting CPU
93.         The data should like this:
94.         'cpuinfo': {'cpuMHz': ['2808.004'],
95.                     'cpucore': ['1'],
96.                     'cpufamily': ['6'],
97.                     'model': ['158'],
98.                     'vendor_id': ['GenuineIntel']}
99.
100.        The target format is like this:
101.        ((cpuMHz', 'cpucore'), ('2808.004',1) ...)
102.        """
103.        temp1=list()
104.        temp2=list()
105.        for item in self.procinfo['cpuinfo']:
106.            temp1.append((item,'10'))
107.            temp2.append(self.procinfo['cpuinfo'][item][0])
108.        self.cpuinfo=(temp1,temp2)
109.        del temp1
110.        del temp2
111.
112.        self.meminfo=((("MemFree", '5'),("MemTotal", '5')), (self.procinfo['me
113.            minfo']["MemFree"][0],self.procinfo['meminfo']["MemTotal"][0]))
114.        """
115.        pidinfo:
116.        {'1': {'Name': ['systemd'],
117.                'PPid': ['0'],
118.                'State': ['S(sleeping)'],
119.                'Threads': ['1']},
120.         '10': {'Name': ['ksoftirqd/0'],
121.                'PPid': ['2'],
122.                'State': ['S(sleeping)'],
123.                'Threads': ['1']},
124.         '1026': {'Name': ['upowerd'],

```

```

125.             'PPid': ['1'],
126.             'State': ['S(sleeping)'],
127.             'Threads': ['3']},
128.             ... ..
129.         }
130.     ...
131.     temp=list()
132.     name= (("PID", '5'), ('Name', '10'), ('PPid', '5'), ("State", '5'), ('Thread
        s', '5'))
133.     temp=[name]
134.     for item in self.procinfo['pidinfo']: # For each process
135.         ca=[item]
136.         for info in name[1:]:
137.             ca.append(self.procinfo['pidinfo'][item][info[0]][0])
138.         temp.append(ca)
139.     self.pidinfo=temp
140.
141.     self.sysinfo=self.procinfo['sysinfo']

```

multibox.py

```

1. from tkinter import *
2. class MultiListbox(Frame):
3.     '''
4.     单行多值 listbox 控
        件, via https://blog.csdn.net/u010039733/article/details/50100499
5.     Usage:
6.     tk = Tk()
7.     Label(tk, text="MultiListbox").pack()
8.     mlb = MultiListbox(tk, (('Subject', 40), ('Sender', 20), ("Date", 10)))
9.     for i in range(1000):
10.         mlb.insert(END, ('Important Message: %d' % i, 'John Doe', '10/10/%4d'
            % (1900+i)))
11.     mlb.pack(expand=YES, fill=BOTH)
12.     tk.mainloop()
13.
14.     '''
15.
16.     def __init__(self, master, lists):
17.         Frame.__init__(self, master)
18.         self.lists = []
19.         for l, w in lists:
20.             frame = Frame(self)
21.             frame.pack(side=LEFT, expand=YES, fill=BOTH)

```



```

22.         Label(frame, text=l, borderwidth=1, relief=RAISED).pack(fill=X)
23.         lb = Listbox(frame, width=w, borderwidth=0, selectborderwidth=0,
24.             relief=FLAT, exportselection=FALSE)
25.         lb.pack(expand=YES, fill=BOTH)
26.         self.lists.append(lb)
27.         lb.bind("<B1-Motion>", lambda e, s=self: s._select(e.y))
28.         lb.bind("<Button-1>", lambda e, s=self: s._select(e.y))
29.         lb.bind("<Leave>", lambda e: "break")
30.         lb.bind("<B2-Motion>", lambda e, s=self: s._b2motion(e.x, e.y))
31.         lb.bind("<Button-2>", lambda e, s=self: s._button2(e.x, e.y))
32.         frame = Frame(self)
33.         frame.pack(side=LEFT, fill=Y)
34.         Label(frame, borderwidth=1, relief=RAISED).pack(fill=X)
35.         sb = Scrollbar(frame, orient=VERTICAL, command=self._scroll)
36.         sb.pack(side=LEFT, fill=Y)
37.         self.lists[0]["yscrollcommand"] = sb.set
38.
39.     def _select(self, y):
40.         row = self.lists[0].nearest(y)
41.         self.selection_clear(0, END)
42.         self.selection_set(row)
43.         return "break"
44.
45.     def _button2(self, x, y):
46.         for l in self.lists:
47.             l.scan_mark(x, y)
48.         return "break"
49.
50.     def _b2motion(self, x, y):
51.         for l in self.lists:
52.             l.scan_dragto(x, y)
53.         return "break"
54.
55.     def _scroll(self, *args):
56.         for l in self.lists:
57.             l.apply(l.yview, args)
58.         return "break"
59.
60.     def curselection(self):
61.         return self.lists[0].curselection()
62.
63.     def delete(self, first, last=None):
64.         for l in self.lists:

```

```
64.         l.delete(first,last)
65.
66.     def get(self, first, last=None):
67.         result = []
68.         for l in self.lists:
69.             result.append(l.get(first,last))
70.         if last:
71.             return apply(map, [None] + result)
72.         return result
73.
74.     def index(self, index):
75.         self.lists[0],index(index)
76.
77.     def insert(self, index, *elements):
78.         for e in elements:
79.             i = 0
80.             for l in self.lists:
81.                 l.insert(index, e[i])
82.                 i = i + 1
83.
84.     def size(self):
85.         return self.lists[0].size()
86.
87.     def see(self, index):
88.         for l in self.lists:
89.             l.see(index)
90.
91.     def selection_anchor(self, index):
92.         for l in self.lists:
93.             l.selection_anchor(index)
94.
95.     def selection_clear(self, first, last=None):
96.         for l in self.lists:
97.             l.selection_clear(first,last)
98.
99.     def selection_includes(self, index):
100.        return self.lists[0].seleciton_includes(index)
101.
102.    def selection_set(self, first, last=None):
103.        for l in self.lists:
104.            l.selection_set(first, last)
```

7.5 实验五

Main.py

```
1. import os
2. from toolbox import *
3. import pickle
4. import logging
5. import commandresolve
6. def console(data:dict,logger):
7.     '''
8.     Main console program
9.     '''
10.    consoleobj=commandresolve.commandresolve(data,logger)
11.    flag=True# to mark if it is time to exit
12.    while (flag):
13.        rawcommand=input(">")
14.        flag=consoleobj.resolvecommand(rawcommand)
15.    #Exit now
16.    logger.info("Exit Successfully")
17.    return #NOTE data should be saved in exit
18.
19.
20. if __name__=="__main__":
21.    #Mainloop
22.    #Search for file
23.    filename="simdisk.bin"
24.    '''
25.    Setup logger
26.    '''
27.    logger = logging.getLogger()#创建对象
28.    logger.setLevel(logging.INFO)#设定起始显示级别
29.
30.    # 创建 Handler
31.    # 终端 Handler
32.    consoleHandler = logging.StreamHandler()
33.    consoleHandler.setLevel(logging.INFO)
34.    # Formatter
35.    formatter = logging.Formatter('%(asctime)s [%(levelname)s] \t %(message)s')
36.    consoleHandler.setFormatter(formatter)
37.
38.
39.    # 添加到 Logger 中
```

```

40.     logger.addHandler(consoleHandler)
41.
42.
43.     if not os.path.isfile(filename):
44.         logger.warning("File not exist. Trying to create...")
45.         createfile(filename,20000000)
46.         with open(filename,'wb') as p:
47.             pickle.dump({},p)
48.     #Build simlink
49.     data=dict()
50.     with open(filename,"rb") as f:
51.         data=pickle.load(f)
52.
53.     if data !={}:
54.         logger.info("Get existed file data, trying to resolve...")
55.         if type(data)!=dict:
56.             print(data)
57.             logger.error("File structure is unable to resolve")
58.
59.
60.         else:
61.             logger.info("File structure is resolved successfully")
62.             logger.info("Jumping to command line...")
63.             console(data,logger)
64.     else:
65.         logger.info("File structure is resolved successfully")
66.         logger.info("Jumping to command line...")
67.         console(data,logger)

```

toolbox.py

```

1. import time
2. import pickle
3. def createfile(filename,size:int):
4.     with open(filename,'wb') as f:
5.         f.seek(size-1)
6.         f.write(b'\x00')
7.
8.
9. def gettime():
10.     return time.strftime('%y-%m-%d %H:%M:%S',time.localtime(time.time()))

```

commandresolve.py

```

1. from fileprocessing import *

```

```
2. from prettytable import PrettyTable
3. import logging
4. from toolbox import *
5. import random
6. import pickle
7. class commandresolve(fileprocessing):
8.     def __init__(self,data:dict,logger,):
9.         self.data=data
10.        self.path=data
11.        self.auth=False
12.        self.logger=logger
13.        self.cmd=['mkfs','ls','pwd','cd','rm','su','mkdir','touch','cat','exit']
14.    def resolvecommand(self,rawcommand):
15.        self.raw=rawcommand
16.        cmdlist=self.raw.split()
17.        if cmdlist[0] not in self.cmd:
18.            self.logger.error("Command not found")
19.            return True
20.        name=cmdlist[0]
21.        if name=='mkfs':
22.            self.mkfs()
23.            return True
24.        elif name=='ls':
25.            self.ls()
26.            return True
27.        elif name=="pwd":
28.            self.pwd()
29.            return True
30.        elif name=="cd":
31.            if len(cmdlist)<2:
32.                self.logger.error("Invalid command")
33.                return True
34.            else:
35.                self.cd(cmdlist[1])
36.                return True
37.        elif name=="rm":
38.            if len(cmdlist)<2:
39.                self.logger.error("Invalid command")
40.                return True
41.            else:
42.                self.rm(cmdlist[1])
43.                return True
44.        elif name=='su':
```

```

45.         self.su()
46.         return True
47.     elif name=="mkdir":
48.         if len(cmdlist)<2:
49.             self.logger.error("Invalid command")
50.             return True
51.         else:
52.             self.mkdir(cmdlist[1])
53.             return True
54.     elif name=='touch':
55.         if len(cmdlist)<2:
56.             self.logger.error("Invalid command")
57.             return True
58.         else:
59.             self.touch(cmdlist[1])
60.             return True
61.     elif name=='cat':
62.         if len(cmdlist)<2:
63.             self.logger.error("Invalid command")
64.             return True
65.         else:
66.             self.cat(cmdlist[1])
67.             return True
68.     elif name=='exit':
69.         self.exit()
70.         return False

```

fileprocessing.py

```

1.  '''Basic processing API'''
2.  from prettytable import PrettyTable
3.  import logging
4.  from toolbox import *
5.  import random
6.  import pickle
7.  class fileprocessing(object):
8.      def __init__(self,data:dict,logger):
9.          self.data=data
10.         self.path=data
11.         self.auth=False
12.         self.logger=logger#logger obj
13.         return
14.     def mkfs(self):
15.         if not self.auth :
16.             self.logger.error("Operation not permitted")

```

```

17.         return
18.     else:
19.         del self.data
20.         del self.path
21.         self.data=dict()
22.         self.path=self.data
23.         self.logger.info("Format successfully")
24.         return
25.     def ls(self):
26.         table = PrettyTable(['Name','Type','Create Time','Change Time','Owner',
27.                               'Size'])
28.         for obj in self.path:
29.             if obj in ['name','type','ctime','mtime','owner','size']:
30.                 continue
31.             table.add_row([obj,self.path[obj]['type'],self.path[obj]['ctime'
32.                               ],self.path[obj]['mtime'],self.path[obj]['owner'],self.path[obj]['size']])
33.         print(table)
34.         return
35.     def pwd(self):
36.         print(self.path)
37.         return
38.     def cd(self,newpath):
39.         if newpath in self.path:
40.             if self.path[newpath]['type']=="Directory":
41.                 self.path=self.path[newpath]
42.                 self.logger.info("Change Successfully.")
43.                 return
44.             else:
45.                 self.logger.error("%s is not a directory"%newpath)
46.                 return
47.         else:
48.             self.logger.error("File or dorectory not exist")
49.             return
50.     def rm(self,target):
51.         if target in self.path:
52.             #if root file
53.             if self.auth:
54.                 del self.path[target]
55.                 self.logger.info("%s has been removed"%target)
56.                 return
57.             elif self.path[target]['owner']=='user' and not self.auth:
58.                 del self.path[target]
59.                 self.logger.info("%s has been removed"%target)
60.                 return

```

```

59.         else:
60.             self.logger.error("Operation is not permitted")
61.             return
62.     else:
63.         self.logger.error("File or dorectory not exist")
64.         return
65.     def su(self):
66.         self.auth= not self.auth
67.         return
68.     def mkdir(self,name):
69.         #Check if the name vaild
70.         if name in ['type','ctime','mtime','owner','size']:
71.             self.logger.error("Name is not permitted")
72.             return
73.         if name in self.path:
74.             self.logger.error("Name already exist")
75.             return
76.         temp=dict()
77.         temp['type']="Directory"
78.         temp['ctime']=gettime()
79.         temp['mtime']=temp['ctime']
80.         temp['owner']=None
81.         temp['size']=None
82.         self.path[name]=temp
83.         return
84.     def touch(self,name):
85.         #Check if the name vaild
86.         if name in ['type','ctime','mtime','owner','size']:
87.             self.logger.error("Name is not permitted")
88.             return
89.         if name in self.path:
90.             self.logger.error("Name already exist")
91.             return
92.         temp=dict()
93.         temp['type']="File"
94.         temp['ctime']=gettime()
95.         temp['mtime']=temp['ctime']
96.         if self.auth:
97.             temp['owner']='root'
98.         else:
99.             temp['owner']='user'
100.        temp['size']=random.randint(0,10000000)
101.        self.path[name]=temp
102.        return

```



```
103.     def cat(self,name):
104.         if name not in self.path:
105.             self.logger.error("File not exist")
106.             return
107.         if self.path[name]['type']=="Directory":
108.             self.logger.error("It is a directory, not a file")
109.             return
110.         if self.auth ==False and self.path[name]['owner']=='root':
111.             self.logger.error("Operation not permitted")
112.             return
113.         self.logger.info("Open file successfully")
114.         return
115.
116.     def exit(self):
117.         with open('simdisk.bin',"wb") as f:
118.             pickle.dump(self.data,f)
119.             self.logger.debug("%s"%self.data)
120.             self.logger.info("Data write successfully")
121.         return
122.
123.
```