華中科技大學

课 程 设 计 报 告

课程名称： 嵌入式操作系统课程设计

专业班级： 物联网 1601

学　　号： U201614897

姓　　名： 潘越

指导教师： 石柯

报告日期： 2018 年 9 月 11 日

计算机科学与技术学院

目录

设计内容一 熟练掌握 Linux 编程环境

1.1 实验目的

（1） 掌握 Linux 操作系统的使用方法。

（2） 熟悉和理解 Linux 编程环境。

1.2 设计内容

（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 实验环境及步骤

1.3.1 实验环境

本次课程设计使用的环境配置如下：

（1）操作系统版本：Arch Linux x86_64

（2）操作系统内核版本：4.17.8

（3）编译器及其版本：GCC version 8.1.1

（4）图形库及其版本：GTK+ 2.0

（5）自动编译工具：CMake version 3.11.4

（6）编程环境：Visual Studio Code

1.3.2 文件拷贝

①首先，文件拷贝应仿照 Linux 自带的 cp 命令，可以做到创建新文件，覆盖目标文件的功能，需要在打开文件的函数 open 中，添加如下的标志：

read_fd = open(argv[1], O_RDONLY);

write_fd = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, get_file_mode(argv[1]);

这些标志的含义如下：

表 1.1 open 内的部分标志含义

标志	含义
O_RDONLY	文件以只读形式打开
O_WRONLY	文件以只写形式打开
O_CREAT	若打开的文件不存在，则创建新文件
O_TRUNC	允许覆盖源文件

利用这些标志，就可以实现标准 cp 命令的功能。

②文件内容拷贝的实现比较简单，在 Linux 下使用函数 read 和 write 读写即可。在这里我又使用了一个缓冲区 buf，对源文件进行一块一块的拷贝，可以提高拷贝的效率。

1.3.3 并发进程演示

①首先配置环境，Arch Linux 已经自带有 GTK+ 2.0 的包，这里我们只需要在 CMake 中包含 GTK+ 2.0 的库。在 CMake 中写入以下几行：

```
# Add GTK2
find_package(PkgConfig REQUIRED)
pkg_check_modules (GTK2 REQUIRED gtk+-2.0)
set(CMAKE_C_STANDARD 11)
include_directories (${GTK2_INCLUDE_DIRS})
link_directories (${GTK2_LIBRARY_DIRS})
add_definitions (${GTK2_CFLAGS_OTHER})
```

这里所做的工作是，找到 GTK+ 2.0 的包，然后设置 C 语言标准为 C11，接着包含 GTK2 的库。

此后若我们编程中需要用到图形库，直接在 CMake 中链接库即可，例如：

```
target_link_libraries(xxx ${GTK2_LIBRARIES})
```

②进程的并发使用 fork()系统调用，若 fork 返回 0，则是子进程，否则是父进程。这样我们可以创建三个进程，并且利用 GTK 创建三个窗口，分别显示要求的内容。

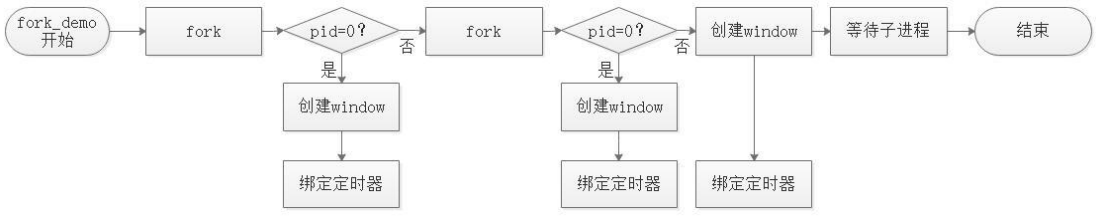

图 1.1 fork_demo 流程图

③利用 gtk，在每个窗口内嵌入一个 label，用 g_strdup_printf 来设置字体格式和打印信息，这样就完成了信息的显示。

④窗口内容刷新功能可以使用 GTK 的定时器 g_timeout_add 实现，每一秒执行一次即可。

1.4 关键代码

（1）文件拷贝核心函数 – do_copy

代码段 1.1 文件拷贝核心函数

```
void do_copy(int read_fd, int write_fd) {
    int read_num;
    char buf[BUF_LEN];
    for (;;) {
        read_num = read(read_fd, buf, BUF_LEN);
        if (read_num == -1)
            err_exit("Copy");
        else if (read_num == 0)
            break;
        write(write_fd, buf, read_num);
    }
}
```

（2）并发进程演示：刷新时间 – refresh_time

代码段 1.2 刷新时间函数

```
gboolean refresh_time(gpointer label) {
    time_t times;
    struct tm *time_buf;
    time(&times);
    time_buf = localtime(&times);

    /* Get now time */
    gchar *text_day = g_strdup_printf("<span
font_desc='48'>%04d-%02d-%02d</span>", \
        1900 + time_buf->tm_year, 1 + time_buf->tm_mon, time_buf->tm_mday);
    gchar *text_time = g_strdup_printf("<span
font_desc='32'>%02d:%02d:%02d</span>", \
        time_buf->tm_hour, time_buf->tm_min, time_buf->tm_sec);
    gchar *text_data = g_strdup_printf("\n%s\n\n%s\n", text_day, text_time);

    gtk_label_set_markup(GTK_LABEL(label), text_data);
    return TRUE;
}
```

三个刷新函数类似，这里就不再列出，详见附录 fork_demo.c 文件。

1.5 调试记录与运行结果

执行如下命令，创建文件夹 build，使用 cmake 工具构建 Makefile，执行 make 命令编译，得到可执行文件 copy 和 fork_demo。

```
mkdir  build
cd  build
cmake  ..
make
```

1.5.1 文件拷贝测试

本次测试的内容如下表：

表 1.2 文件拷贝测试内容

项目	测试	测试结果
1	测试复制文件到空文件	通过
2	测试复制并创建文件	通过
3	测试复制并覆盖一个其他类型文件	通过

（1）测试项目 1

创建 test 文件夹，将 copy 程序复制进来，并测试复制本课程设计的指导文档，先使用 touch 创建一个空文件，接着执行复制，命令如下图：

```
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  cp copy ../../../test
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  cd ../../../test
panyue@Saltedfish ~/code/test  ls
assignments.ppt  copy
panyue@Saltedfish ~/code/test  touch test.ppt
panyue@Saltedfish ~/code/test  ls
assignments.ppt  copy  test.ppt
panyue@Saltedfish ~/code/test  ./copy assignments.ppt test.ppt
panyue@Saltedfish ~/code/test
```

图 1.2 测试项目 1 命令

复制完毕后，打开复制前和后的文件对比，验证程序的正确性。

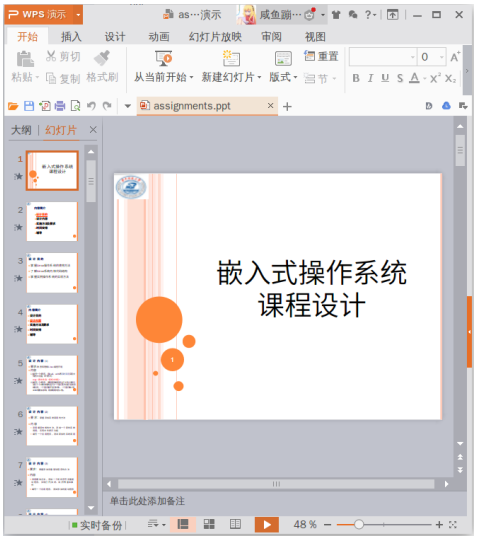

图 1.3 复制前文档

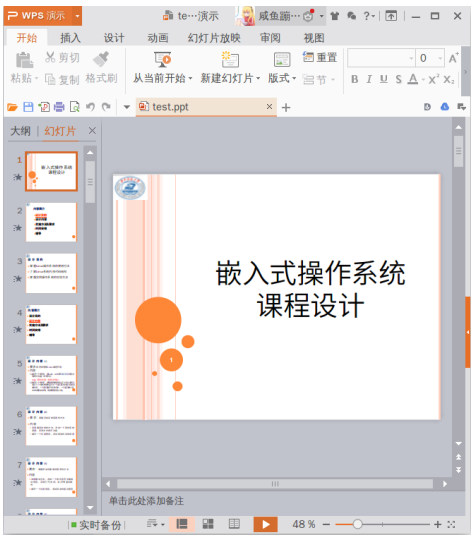

图 1.4 复制后文档

（2）测试项目 2

删除刚才复制的文件 test.ppt，测试复制并创建文件，命令如下图：

```
panyue@Saltedfish ~/code/test rm test.ppt
panyue@Saltedfish ~/code/test ls
assignments.ppt  copy
panyue@Saltedfish ~/code/test ./copy assignments.ppt test.ppt
panyue@Saltedfish ~/code/test ls
assignments.ppt  copy  test.ppt
panyue@Saltedfish ~/code/test
```

图 1.5 测试项目 2 命令

可以看到，文件仍然被正常复制，并且打开后仍然一致。

（3）测试项目 3

删除上一次复制的文件，并且创造一个 copy 的复制，测试复制文档到可执行文件。

```
panyue@Saltedfish ~/code/test ls
assignments.ppt  copy  test.ppt
panyue@Saltedfish ~/code/test rm test.ppt
panyue@Saltedfish ~/code/test ls
assignments.ppt  copy
panyue@Saltedfish ~/code/test ./copy copy exe
panyue@Saltedfish ~/code/test ls
assignments.ppt  copy  exe
panyue@Saltedfish ~/code/test ./copy assignments.ppt exe
panyue@Saltedfish ~/code/test ls
assignments.ppt  copy  exe
panyue@Saltedfish ~/code/test
```

图 1.6 测试项目 3 命令

由于是可执行文件，无法直接打开，我们对 exe 右键，选择使用 wps 演示打开，可以看到和原文档一样的 ppt，验证了程序的正确性。

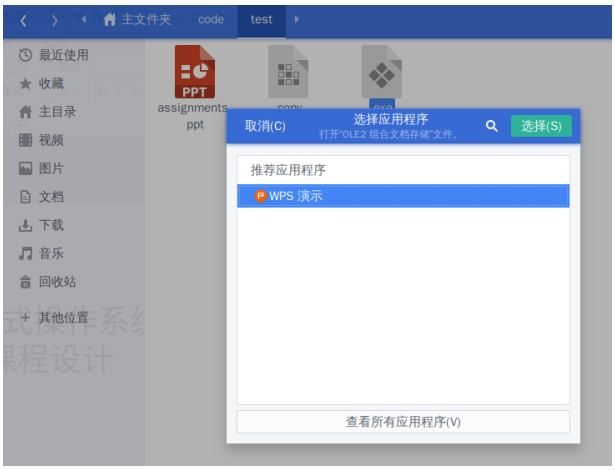

图 1.7 使用 wps 演示打开复制的可执行文件

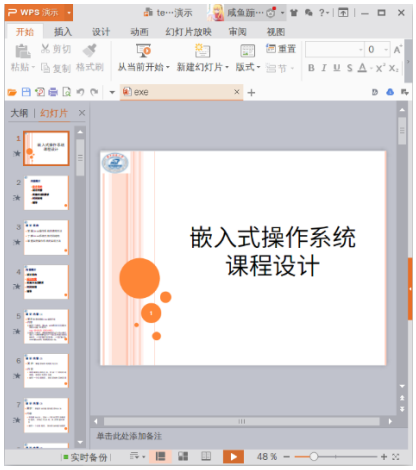

图 1.8 打开后的文件

1.5.2 并发演示

直接运行./fork_demo，可以得到如下图的结果，观察三个窗口，可以看到一秒一次的刷新内容，验证我们的测试通过。

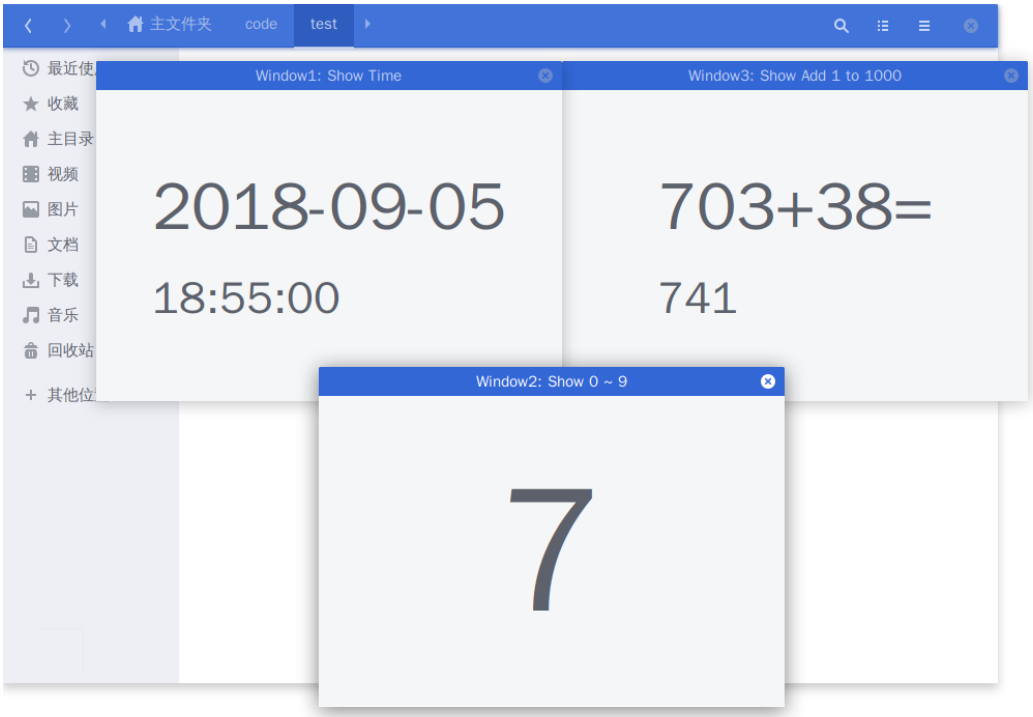

图 1.9 并发演示

设计内容二 增加系统调用

2.1 实验目的

（1）了解 Linux 系统内核代码结构。

（2）掌握添加系统调用的方法。

2.2 设计内容

（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 实验环境及步骤

2.3.1 实验环境

本次课程设计使用的环境配置如下：

（1）操作系统版本：Arch Linux x86_64

（2）操作系统内核版本：4.17.8

（3）编译器及其版本：GCC version 8.1.1

（4）自动编译工具：CMake version 3.11.4

（5）编程环境：Visual Studio Code

2.3.2 内核编译

（1）Linux 系统调用的原理

用户进程不能访问内核所占内存空间，也不能调用内核函数。进程调用一个特殊的指令，这个指令会跳到一个事先定义的内核中的一个位置。在 Intel CPU 中，由中断 INT 0x80 实现。 (与 DOS 功能调用 int0x21 很相似)跳转到的内核位置叫做 sysem_call。检查系统调用号，这个号码代表进程请求哪种服务。然后，它查看系统调用表(sys_call_table)找到所调用的内核函数入口地址。接着，就调用函数，等返回后，做一些系统检查，最后返回到进程（如果这个进程时间用尽，就返回到其他进程）。

（2）内核编译步骤

由于本次实验使用的环境为 Arch Linux 和当时最新版本的内核 4.17.8，所以在内核编译与系统调用增加的步骤上与在传统的 Ubuntu 16.06 以及内核版本小于 4.15 的平台上实验的操作步骤有很大的区别，这里为我在实验中通过查阅资料以及阅读内核源码摸索出的步骤。在设计二上还是花了不少时间的。

①获取内核源码

执行如下命令，从官网上下载 linux-4.17.8 的源码，并解压缩。

wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.17.8.tar.xz
tar -xvf linux-4.17.8.tar.xz linux-4.17.8

②在 linux-4.17.8/arch/x86/entry/syscalls/syscall_64.tbl 文件中的 64 位系统调用最下面，增加如下的三行，添加三个系统调用入口。

```
330 common  pkey_alloc    __x64_sys_pkey_alloc
331 common  pkey_free     __x64_sys_pkey_free
332 common  statx         __x64_sys_statx
333 common  testcall      __x64_sys_testcall
334 common  teststr       __x64_sys_teststr
335 common  mycopy        __x64_sys_mycopy
```

图 2.1 添加系统调用入口

③在 linux-4.17.8/kernel/sys.c 文件中，添加系统调用实现的代码。注意，在 Arch Linux 环境下必须使用 SYSCALL_DEFINE 系列的宏来实现系统调用。

```c
SYSCALL_DEFINE0(testcall) {
  printk(KERN_INFO "Hello world!");
  return 0;
}

SYSCALL_DEFINE1(teststr, const char *, str) {
  char buf[256];
  long read_num;

  /* Set memory access range */
  mm_segment_t fs = get_fs();
  set_fs(get_ds());

  /* Copy str form user space to kernel space */
  read_num = strncpy_from_user(buf, str, sizeof(buf));
  if (read_num < 0 || read_num == sizeof(buf)) {
    set_fs(fs);
    return -EFAULT;
  }

  printk(KERN_INFO "System call teststr: %s", buf);
  set_fs(fs);
  return 0;
}

SYSCALL_DEFINE2(mycopy, const char *, s_file, const char *, t_file) {
  struct kstat k_buf;
  char copy_buf[1024];
  char s_filename[256], t_filename[256];
  int read_fd, write_fd;
  long read_num;
```

图 2.2 添加系统调用实现

④在内核源码目录下，执行如下命令，导出正在运行的内核的配置，用于新内核的编译。

zcat /proc/config.gz > .config

⑤在.config 文件内，修改新内核的后缀为自己的标记，这里我把 CONFIG_LOCALVERSION 的值改成了-zxcpyp.

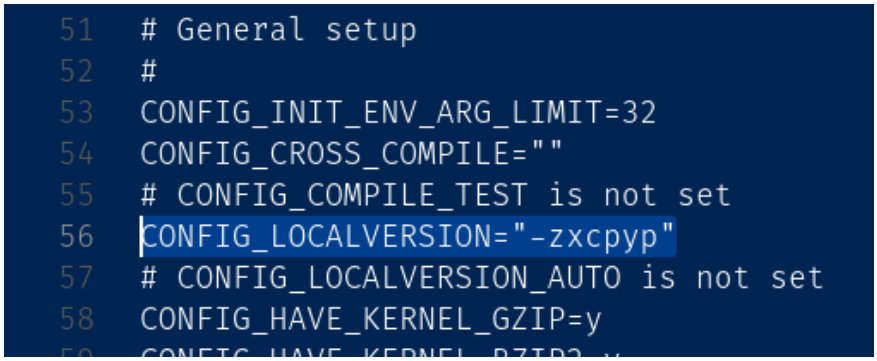

图 2.3 修改内核版本说明

⑥执行 make -j8，执行编译。-j 后面的数字代表编译时开启的线程数量，由于我的电脑是八核的，所以这里编译使用 8 个线程。接着编译模块，执行 make modules_install.

⑦拷贝生成的内核镜像至/boot 目录，这里我是 x86_64 的系统架构，如果是 32 位的则应该拷贝到 x86 目录下。

cp arch/x86_64/boot/bzImage /boot/vmlinuz-linux-zxcpyp

⑧制作初始化内存盘，复制和修改 mkinitcpio preset，就能用官方内核一样的方式生成自定义内核的 initramfs 镜像。

sed s/linux/linux-zxcpyp/g </etc/mkinitcpio.d/linux.preset >/etc/mkinitcpio.d/linux-zxcpyp.preset

mkinitcpio -p linux-zxcpyp

⑨更新 grub，这样使得我们在重启的时候可以看到新内核的入口，从而进入我们的新内核。

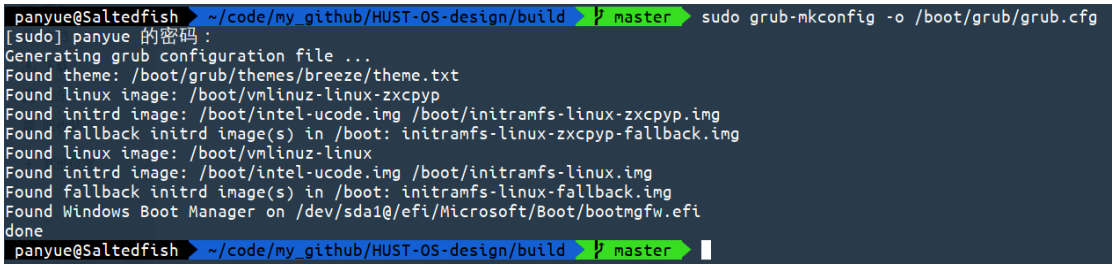

图 2.4 更新 grub

⑩进入新内核

重启之后，选择 Advanced options for Arch Linux，就可以看见我们新添加的内核入口，选择 Arch Linux, Linux linux-zxcpyp 即可进入新内核。

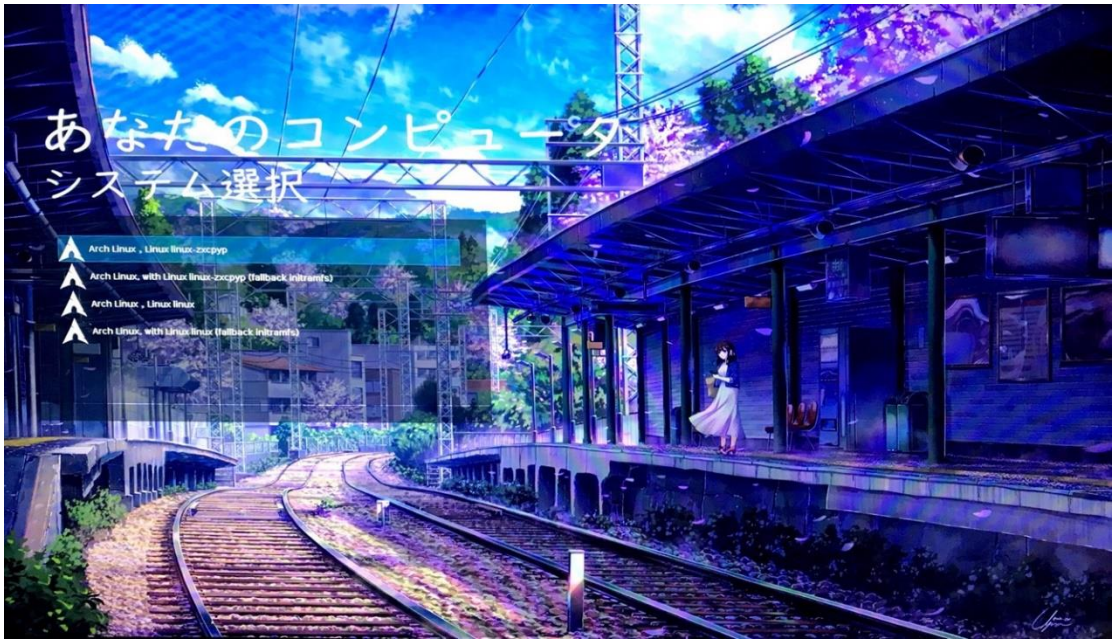

图 2.5 grub 进入新内核

2.3.3 显卡修复

我在编译完内核后，重新启动进入新内核的时候，出现了无法打开图形界面的问题，经过在 stackoverflow 上查找问题之后，发现是显卡模块不适配。需要在重启之后切换进字符界面的一个 tty，执行 NVIDIA 提供的脚本，重新编译显卡模块，才可以启动图形界面管理器，打开我的 GNOME 图形界面。以下是修复的步骤

①查看自己的显卡类型，我的是 GTX 1060

```
panyue@Saltedfish ~ lspci -k | grep -A 2 -E "(VGA|3D)"
01:00.0 VGA compatible controller: NVIDIA Corporation GP106M [GeForce GTX 1060 Mobile] (rev a1)
        Subsystem: Device 1d05:1030
        Kernel driver in use: nvidia
panyue@Saltedfish ~
```

图 2.6 更新查看显卡版本

②在 NVIDIA 官网 https://www.nvidia.com/Download/index.aspx 上获取对应的脚本，并在编译完内核重启后执行。

./NVIDIA-Linux-x86_64-390.77.run

reboot

再次重启后即可正常进入系统图形界面了。

2.3.4 编写测试程序

对应我增加的三个系统调用（打印 hello world，打印传递的字符串参数，文件拷贝），编写三个测试程序完成测试。

注意的是，在编写测试时，调用新增的系统调用需要用 syscall 函数和及其

系统调用号来调用，例如调用 mycopy 函数如下：

```
syscall(335, argv[1], argv[2]);
```

2.4 关键代码

（1）系统调用 3 – mycopy

代码段 2.1 系统调用 – 文件拷贝

```
SYSCALL_DEFINE2(mycopy, const char *, s_file, const char *, t_file) {
        struct kstat k_buf;
        char copy_buf[1024];
        char s_filename[256], t_filename[256];
        int read_fd, write_fd;
        long read_num;

        /* Set memory access range */
        mm_segment_t fs = get_fs();
        set_fs(get_ds());

        /* Get source and target file name */
        read_num = strncpy_from_user(s_filename, s_file, sizeof(s_filename));
        if (read_num < 0 || read_num == sizeof(s_filename)) {
                set_fs(fs);
                return -EFAULT;
        }
        read_num = strncpy_from_user(t_filename, t_file, sizeof(t_filename));
        if (read_num < 0 || read_num == sizeof(t_filename)) {
                set_fs(fs);
                return -EFAULT;
        }

        /* Get source file mode */
        if (vfs_stat(s_filename, &k_buf) != 0) {
                set_fs(fs);
                return -EFAULT;
        }

        /* Open files */
        if ((read_fd = ksys_open(s_filename, O_RDONLY, S_IRUSR)) == -1) {
                set_fs(fs);
                return -EFAULT;
        }
        printk("After open source file");
```

```
        if ((write_fd = ksys_open(t_filename, O_WRONLY | O_CREAT |
O_TRUNC, k_buf.mode)) == -1) {
                set_fs(fs);
                return -EFAULT;
        }

        /* Do copy */
        for (;;) {
                read_num = ksys_read(read_fd, copy_buf, sizeof(copy_buf));
                if (read_num < 0) {
                        set_fs(fs);
                        return -EFAULT;
                } else if (read_num == 0)
                        break;
                ksys_write(write_fd, copy_buf, read_num);
        }

        ksys_close(read_fd);
        ksys_close(write_fd);

        /* Restore previous memory access */
        set_fs(fs);

        return 0;
}
```

在编写该系统调用时，需要注意的一点是，在调用 ksys_read 等函数时，需要暂时将访问限制值设置为内核的内存访问范围，待程序结束前再修改为原来的值。否则会导致内存保护检查错误。

（2）系统调用测试 – testcp

代码段 2.2 系统调用测试 – 文件拷贝

```
int main(int argc, char **argv) {
   printf("/*\n");
   printf(" * HUST OS Design - Part II\n");
   printf(" *\n");
   printf(" * Test03: Test mycopy\n");
   printf(" */\n\n");
   if (argc != 3) {
      printf("Usage ./testcp <src> <dst>\n");
```

```
    return 0;
  }
  printf("Copy: %s -> %s\n", argv[1], argv[2]);
  long ret = syscall(335, argv[1], argv[2]);
  printf("ret: %ld\n", ret);
  printf("errno: %d\n", errno);
  return 0;
}
```

2.5 调试记录与运行结果

在 build 文件夹下执行 make 后，可以看到可执行程序 testcall、teststr 和 testcp，分别是测试设计二添加的三个系统调用的。

本次测试的内容如下表：

表 2.1 系统调用测试内容

项目	测试	测试结果
1	测试系统调用打印字符串	通过
2	测试系统调用传参	通过
3	测试系统调用复制文件	通过

（1）测试项目 1&2

在命令行分别执行 testcall 和 teststr 三次如下图：

```
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master ./testcall
/*
 * HUST OS Design - Part II
 *
 * Test01: Test syscall
 * issue: dmesg to see "Hello world!"
 */
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master ./testcall
/*
 * HUST OS Design - Part II
 *
 * Test01: Test syscall
 * issue: dmesg to see "Hello world!"
 */
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master ./testcall
/*
 * HUST OS Design - Part II
 *
 * Test01: Test syscall
 * issue: dmesg to see "Hello world!"
 */
```

图 2.7 测试 testcall

```
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build master ./teststr asdfhdfghdf
/*
 * HUST OS Design - Part II
 *
 * Test02: Test string pass
 * issue: dmesg to see your input string
 */

Arg: asdfhdfghdf
ret: 0
errno: 0
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build master ./teststr qwetrwerh
/*
 * HUST OS Design - Part II
 *
 * Test02: Test string pass
 * issue: dmesg to see your input string
 */

Arg: qwetrwerh
ret: 0
errno: 0
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build master ./teststr wetyeqrthgf
/*
 * HUST OS Design - Part II
 *
 * Test02: Test string pass
 * issue: dmesg to see your input string
 */

Arg: wetyeqrthgf
ret: 0
errno: 0
```

图 2.8 测试 teststr

接着执行 dmesg，可以看到 printk 执行的结果。

```
[ 1863.135898] Hello world!
[ 1864.248714] Hello world!
[ 1865.006249] Hello world!
[ 1921.823788] System call testtstr: asdfhdfghdf
[ 1925.065559] System call testtstr: qwetrwerh
[ 1930.505870] System call testtstr: wetyeqrthgf
```

图 2.9 printk 打印结果

（3）测试项目 3

执行 testcp，复制课程设计的文档，得到结果如下，验证了程序的正确性。

```
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build master ./testcp ~/Assignment.ppt ~/test.ppt
/*
 * HUST OS Design - Part II
 *
 * Test03: Test mycopy
 */

Copy: /home/panyue/Assignment.ppt -> /home/panyue/test.ppt
ret: 0
errno: 0
```

图 2.10 测试 testcp

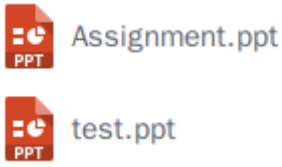

图 2.10 复制结果

设计内容三 增加设备驱动程序

3.1 实验目的

（1）了解 Linux 系统内核代码结构。

（2）掌握添加设备驱动程序的方法。

3.2 设计内容

（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 实验环境及步骤

3.3.1 实验环境

本次课程设计使用的环境配置如下：

（1）操作系统版本：Arch Linux x86_64

（2）操作系统内核版本：4.17.8

（3）编译器及其版本：GCC version 8.1.1

（4）自动编译工具：CMake version 3.11.4

（5）编程环境：Visual Studio Code

3.3.2 设备驱动实现

（1）Linux 设备驱动

Linux 内核中的设备驱动程序是一组常驻内存的具有特权的共享库，是低级硬件处理例程。对用户程序而言，设备驱动程序隐藏了设备的具体细节，对各种不同设备提供了一致的接口，一般来说是把设备映射为一个特殊的设备文件，用户程序可以象对其它文件一样对此设备文件进行操作。

Linux 支持 3 种设备：字符设备、块设备和网络设备。

设备由一个主设备号和一个次设备号标识。主设备号唯一标识了设备类型，即设备驱动程序类型，它是块设备表或字符设备表中设备表项的索引。次设备号仅由设备驱动程序解释,一般用于识别在若干可能的硬件设备中，I/O 请求所涉及到的那个设备。

一个典型的驱动程序,大体上可以分为这么几个部分:

①注册设备：

在系统初启,或者模块加载时候,必须将设备登记到相应的设备数组,并返回设备的主设备号；

②定义功能函数：

对于每一个驱动函数来说，都有一些和此设备密切相关的功能函数。以最常

用的块设备或者字符设备来说，都存在着诸如 open()、read()这一类的操作。当系统调用这些调用时，将自动的使用驱动函数中特定的模块。来实现具体的操作；

③卸载设备：

在不用这个设备时,可以将它卸载，主要是从/dev 中取消这个设备的特殊文件。

（2）设备驱动实现

①功能设计

首先设计设备驱动功能，这里我将其设计为一个大小为 8KB 大小的字符设备，并且实现如下图几个功能：

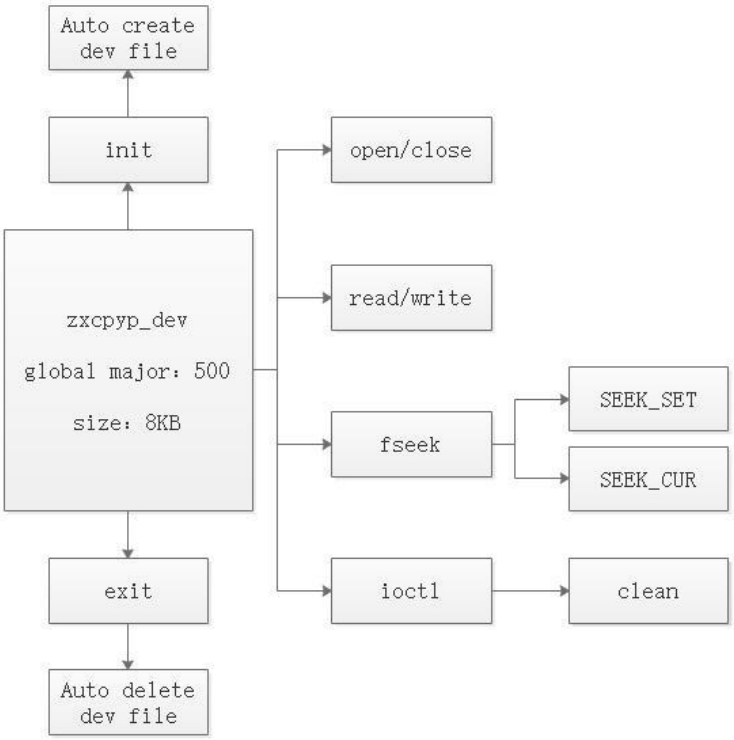

图 3.1 设备驱动设计

其中，在加载和退出时实现自动创建和删除设备文件，无需再使用 mknod 等命令。由于该字符设备的大小限制为 8KB 了，fseek 命令只实现了 SEEK_SET 与 SEEK_CUR。此外，清空设备的 clean 命令使用通用接口 ioctl 实现。

②数据结构设计

将字符设备的 cdev 结构体和 8KB 的数据块整体作为一个结构体，并声明了一个全局变量指针，供其他函数使用。

```
struct zxcpyp_dev {
  struct cdev cdev;
  unsigned char mem[ZXCPYPMEM_SIZE];
```

```
};
static struct zxcpyp_dev *zxcpyp_devp;
```

③需要实现的函数原型

在设计三中需要实现的各个子函数原型如下表：

表 3.1 需要实现的各个子函数原型

入口点	函数原型	功能
.open	static int zxcpypdriver_open(struct inode *inodep, struct file *filep);	打开设备
.release	static int zxcpypdriver_release(struct inode *inodep, struct file *filep)	关闭设备
.read	static ssize_t zxcpypdriver_read(struct file *filep, char __user *buf, size_t count, loff_t *offset)	从设备中读数据
.write	static ssize_t zxcpypdriver_write(struct file *filep, const char __user *buf, size_t count, loff_t *offset)	向设备中写数据
.llseek	static loff_t zxcpypdriver_llseek(struct file *filep, loff_t offset, int whence)	更改设备当前的偏移量
.unlocked_ioctl	static long zxcpypdriver_ioctl(struct file *filep, unsigned int cmd, unsigned long arg)	通用接口，这里实现清空设备

④接口注册

在 file_operations 结构体 fops 中存在着大量的函数入口点，在这里指定我们要实现的各个函数如下：

```
static const struct file_operations fops = {
   .owner = THIS_MODULE,
   .open = zxcpypdriver_open,
   .release = zxcpypdriver_release,
   .read = zxcpypdriver_read,
   .write = zxcpypdriver_write,
   .llseek = zxcpypdriver_llseek,
   .unlocked_ioctl = zxcpypdriver_ioctl,
};
```

⑤打开/关闭函数设计

打开和关闭的实现是最简单的，打开设备只需要将 file 指针的私有数据 private_data 指向设备结构体 zxcpyp_dev 即可。关闭设备可以直接关闭。

⑥读/写函数设计

读写的实现稍微复杂一些，这里要分情况讨论。如果剩余空间足够的话，可以读/写指定的长度，如果空间不够的话，就只能读/写剩余的空间内的数据了。在完成操作后还要修改文件偏移量。

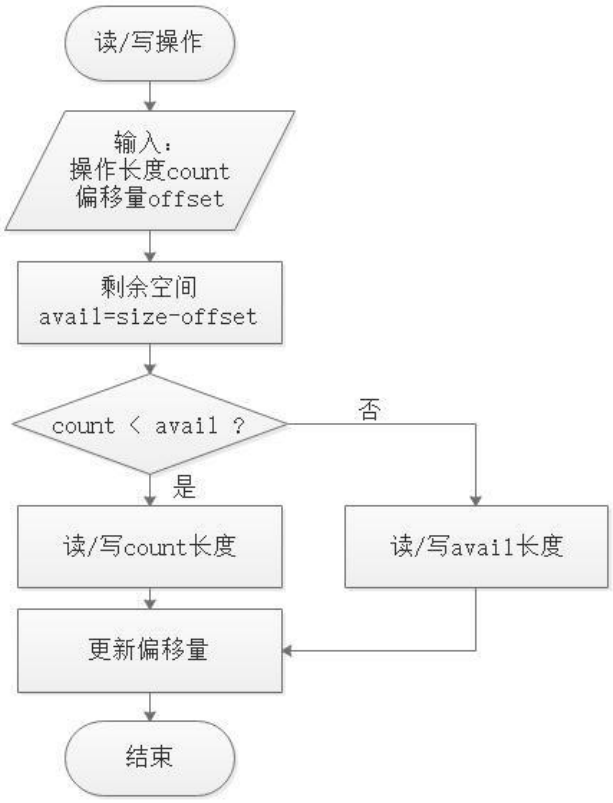

图 3.2 读写读/写函数设计

⑦fseek 函数设计

fseek 函数的核心就是更改偏移量，直接根据输入和当前的偏移量来计算新的偏移量即可，注意超过范围的操作应该不被允许。

表 3.2 fseek 函数功能

标志	功能	实现
SEEK_SET	设置文件偏移量到从文件开头开始的某个位置	设置 filep->f_pos 为指定的位置
SEEK_CUR	设置文件偏移量从文件当前位置开始的某个位置	设置 filep->f_pos 为指定的位置+ offset
SEEK_END	设置文件偏移量到从文件结尾开始的某个位置	本设备中无法实现，因为该字符设备大小已经固定

⑧clean 功能设计

clean 功能的实现采用了通用接口 ioctl，通过宏定义定义了一个代表 clean 操作的数值，在 ioctl 中通过 memset 函数实现清零。

⑨自动生成和删除设备文件

在 init 函数和 exit 函数中，利用 class 和 device 结构体实现该功能（详见 3.4 关键代码）

（3）编译和加载内核模块

在./PartIII-Device_driver/zxcpyp_dev 文件夹中，创建如下的 Makefile 文件，用于编译设备驱动程序。

代码段 3.1 设备驱动 Makefile

```
# Kernel lib
KVERS :=$(shell uname -r)

# Path
PWD :=$(shell pwd)

# Build modules
obj-m += zxcpyp_dev.o

build:
	make -C /lib/modules/$(KVERS)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(KVERS)/build M=$(PWD) clean
```

编译完成后，在目录下执行以下命令，加载内核模块。同时可以自动生成设备文件。

sudo insmode zxcpyp_dev.ko

执行 ls /dev，可以看到生成的设备文件（最右侧），加载完成。

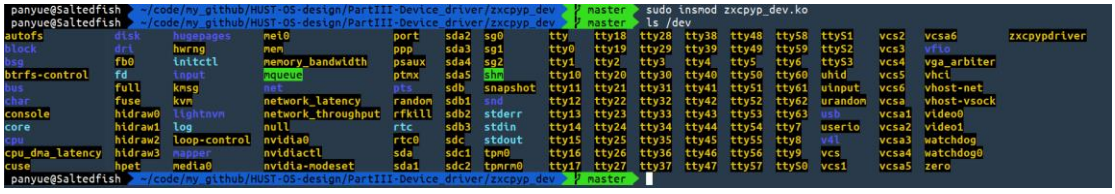

图 3.3 设备加载结果

3.3.3 编写测试程序

测试应做到可以覆盖所有的操作，这里我在测试文件中实现了三个功能：

1.在设备第 m 个位置开始写 n 长度的字符串

2.从设备第 m 个位置开始读 n 长度的字符串

3.清空设备

通过以上三个功能的正确性即可验证 read，write，fseek，ioctl 的实现的正确性。

3.4 关键代码

（1）从设备读数据 – zxcpypdriver_read

代码段 3.2 设备驱动 – 读数据

```
static ssize_t zxcpypdriver_read(struct file *filep, char __user *buf, size_t count,
loff_t *offset) {
    printk(KERN_INFO "ZXCPYP Driver: start read\n");

    int ret = 0;
    size_t avail = ZXCPYPMEM_SIZE - *offset;
    struct zxcpyp_dev *dev = filep->private_data;

    /* Available memory exists */
    if (count <= avail) {
        if (copy_to_user(buf, dev->mem + *offset, count) != 0)
            return -EFAULT;
        *offset += count;
        ret = count;
    }
    /* Available memory not enough */
    else {
        if (copy_to_user(buf, dev->mem + *offset, avail) != 0)
            return -EFAULT;
        *offset += avail;
        ret = avail;
    }

    printk(KERN_INFO "ZXCPYP Driver: read %u bytes\n", ret);
    return ret;
}
```

（2）修改偏移量 – zxcpypdriver_llseek

代码段 3.3 设备驱动 – 修改偏移量

```
static loff_t zxcpypdriver_llseek(struct file *filep, loff_t offset, int whence) {
    printk(KERN_INFO "ZXCPYP Driver: start llseek\n");

    loff_t ret = 0;
    switch (whence) {
    /* SEEK_SET */
    case 0:
        if (offset < 0) {
```

```
        ret = -EINVAL;
        break;
    }
    if (offset > ZXCPYPMEM_SIZE) {
        ret = -EINVAL;
        break;
    }
    ret = offset;
    break;
/* SEEK_CUR*/
case 1:
    if ((filep->f_pos + offset) > ZXCPYPMEM_SIZE) {
        ret = -EINVAL;
        break;
    }
    if ((filep->f_pos + offset) < 0) {
        ret = -EINVAL;
        break;
    }
    ret = filep->f_pos + offset;
    break;
/*
  * SEEK_END: Here we can't use SEEK_END,
  *                 beacuse the memory is solid.
  *
case 2:
    if (offset < 0) {
        ret = -EINVAL;
        break;
    }
    ret = ZXCPYPMEM_SIZE + offset;
    break;*/
/* Else: return error */
default:
    ret = -EINVAL;
}

if (ret < 0)
    return ret;

printk(KERN_INFO "ZXCPYP Driver: set offset to %u\n", ret);
filep->f_pos = ret;
```

```
    return ret;
}
```

（3）设备初始化 – zxcpypdriver_init

代码段 3.3 设备驱动 – 修改偏移量

```
static int __init zxcpypdriver_init(void) {
    printk(KERN_INFO "Load module: zxcpypdriver\n");

    int ret;
    dev_t devno = MKDEV(GLOBAL_MAJOR, 0);
    ret = register_chrdev_region(devno, 1, "zxcpypdriver");
    if (ret < 0) {
        printk(KERN_ALERT "Registering the character device failed with %d\n",
ret);
        return ret;
    }

    /* Alloc memory for device */
    zxcpyp_devp = kzalloc(sizeof(struct zxcpyp_dev), GFP_KERNEL);
    if (zxcpyp_devp == NULL) {
        printk(KERN_ALERT "Alloc memory for device failed\n");
        ret = -ENOMEM;
        goto failed;
    }
    memset(zxcpyp_devp->mem, 0, ZXCPYPMEM_SIZE);

    /* Setup device */
    cdev_init(&zxcpyp_devp->cdev, &fops);
    zxcpyp_devp->cdev.owner = THIS_MODULE;
    cdev_add(&zxcpyp_devp->cdev, devno, 1);

    /* Creat device file */
    class = class_create(THIS_MODULE, "zxcpypdriver");
    if (IS_ERR(class)) {
        ret = PTR_ERR(class);
        printk(KERN_ALERT "Creat class for device file failed with %d\n", ret);
        goto failed;
    }
    device = device_create(class, NULL, devno, NULL, "zxcpypdriver");
    if (IS_ERR(device)) {
        class_destroy(class);
```

```
        ret = PTR_ERR(device);
        printk(KERN_ALERT "Creat device file failed with %d\n", ret);
        goto failed;
    }

    return 0;

  failed:
    unregister_chrdev_region(devno, 1);
    return ret;
}
```

3.5 调试记录与运行结果

在 build 文件夹下执行 make 后，可以看到可执行程序 testdev，为测试设备驱动的程序。

本次测试的内容如下表：

表 3.3 设备驱动测试内容

项目	测试	命令	理论上的结果	测试结果
1	测试向设备写数据	./testdev write 0 “Hello world\!”	Hello world!（设备）	通过
2	查看设备内容	cat /dev/zxcpypdriver	Hello world! （设备）	通过
3	测试向设备写数据	./testdev write 7 “ABCDEFG”	Hello wABCDEFG（设备）	通过
4	查看设备内容	cat /dev/zxcpypdriver	Hello wABCDEFG（设备）	通过
5	测试从设备读数据	./testdev read 0 99	Hello wABCDEFG（输出）	通过
6	测试从设备读数据	./testdev read 3 6	lo w（设备）	通过
7	测试从设备读数据	./testdev read 8 2	BC（设备）	通过
8	测试清空设备	./testdev ioctl clear	无（设备）	通过
9	查看设备内容	cat /dev/zxcpypdriver	无（输出）	通过
10	测试从设备读数据	./testdev read 0 99	无（输出）	通过

（1）测试项目 1&2

命令的执行和结果如下图：

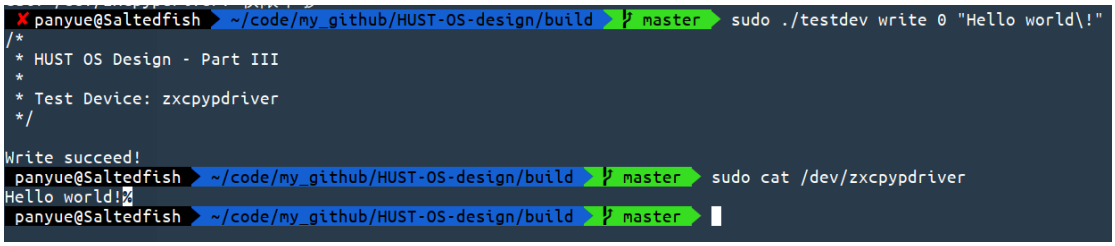

图 3.4 测试项目 1&2

（2）测试项目 3&4

命令的执行和结果如下图：

```
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  sudo ./testdev write 7 "ABCDEFG"
/*
 * HUST OS Design - Part III
 *
 * Test Device: zxcpypdriver
 */

Write succeed!
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  sudo cat /dev/zxcpypdriver
Hello wABCDEFG%
```

图 3.5 测试项目 3&4

（3）测试项目 5&6&7

命令的执行和结果如下图：

```
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  sudo ./testdev read 0 99
/*
 * HUST OS Design - Part III
 *
 * Test Device: zxcpypdriver
 */

Read: Hello wABCDEFG
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  sudo ./testdev read 3 6
/*
 * HUST OS Design - Part III
 *
 * Test Device: zxcpypdriver
 */

Read: lo wAB
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  sudo ./testdev read 8 2
/*
 * HUST OS Design - Part III
 *
 * Test Device: zxcpypdriver
 */

Read: BC
```

图 3.6 测试项目 5&6&7

（4）测试项目 8&9&10

```
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  sudo ./testdev ioctl clear
/*
 * HUST OS Design - Part III
 *
 * Test Device: zxcpypdriver
 */

Clear success!
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  sudo cat /dev/zxcpypdriver
panyue@Saltedfish ~/code/my_github/HUST-OS-design/build  master  sudo ./testdev read 0 99
/*
 * HUST OS Design - Part III
 *
 * Test Device: zxcpypdriver
 */

Read:
```

图 3.7 测试项目 8&9&10

十项测试全部通过，验证了设备驱动的正确性。

设计内容四 系统监控器

4.1 实验目的

（1）掌握实例操作系统的实现方法。

（2）使用 GTK/QT 实现一个系统监控器。

4.2 设计内容

（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)。

4.3 实验环境及步骤

4.3.1 实验环境

本次课程设计使用的环境配置如下：

（1）操作系统版本：Arch Linux x86_64

（2）操作系统内核版本：4.17.8

（3）编译器及其版本：GCC version 8.1.1

（4）图形库及其版本：GTK+ 2.0

（5）自动编译工具：CMake version 3.11.4

（6）编程环境：Visual Studio Code

4.3.2 基本知识

（1）/porc 文件系统

/proc 文件系统是一个虚拟文件系统，通过它可以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。

/proc 文件系统存在于系统的/proc 目录下，与其它常见的文件系统不同的是，它并不是真正意义上的文件系统，它存在于内存中，并不占用磁盘空间，它包含一些结构化的目录和虚拟文件，向用户呈现内核中的一些信息，也可以用作一种从用户空间向内核发送信息的手段。

这些虚拟文件使用查看命令查看时会返回大量信息，但文件本身的大小却会显示为 0 字节。此外，这些特殊文件中大多数文件的时间及日期属性通常为当前系统时间和日期。

现在系统中常用的命令，比如 ps，top 等都是由/proc 文件系统实现的。本设计即在/proc 文件系统下读数据，并将其以直观形式呈现。

（2）GTK+

GTK+(GIMP Toolkit) 是一套在 GIMP 的基础上发展而来的高级的、可伸缩的现代化、跨平台图形工具包，提供一整套完备的图形构件，适用于大大小小各种软件工程项目，不论是小到只需要一个窗口，还是复杂得如桌面环境。简单来说，GTK+ 是一种函数库是用来帮助制作图形交互界面的。同时，它遵循 LGPL 许可证，所以用户可以用它来开发开源软件、自由软件，甚至是封闭源代码的商业软件，而不用花费任何钱来购买许可证和使用权。

由于 GTK+ 整个函数库都是由 C 语言来编写的，因此极适合为 C 语言编写的程序提供 GUI，因此本设计选用了 GTK+ 2.0 作为图形库。

常用的控件有 window，notebook，hbox，vbox，label，frame 等等。

4.3.3 系统监视器实现

（1）总体设计

系统监视器利用 notebook 控件设计了六个页面，如下图：

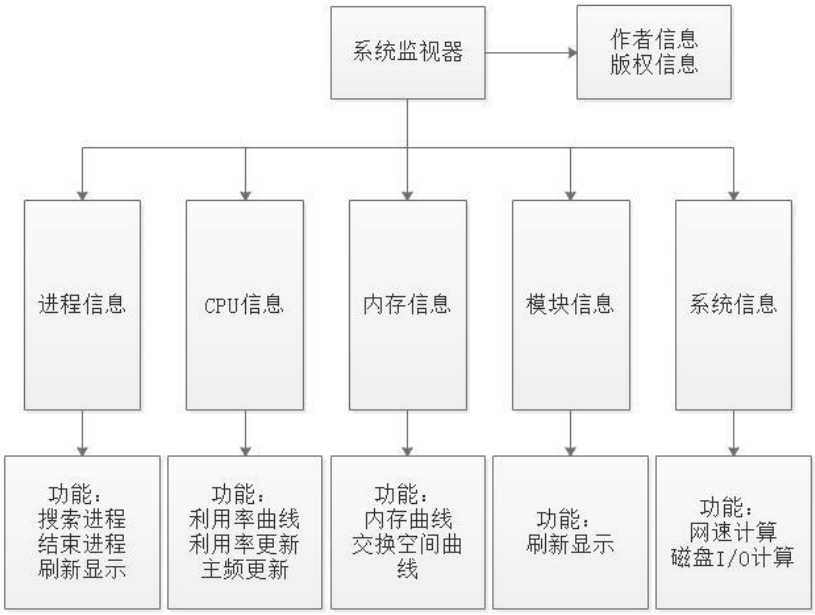

图 4.1 系统监视器设计

（2）系统数据获取

①进程详细信息

/proc 文件系统中存在着很多以数字为文件名的文件夹，这些文件夹包含的就是对应进程号的详细信息，其中我们要获取进程的详细信息被存放在 status 和 stat 两个文件内。

他们的区别是 status 比较直观，按行列出了进程的每个属性和其对应的值，而 stat 则是只有数字，每个一个空格一个。

```
panyue@Saltedfish  /proc/1  cat stat
1 (systemd) S 0 1 1 0 -1 4194560 8375 56126 90 305 17 80 208 554 20 0 1 0 2 240078848 2211 184467440
73709551615 1 1 0 0 0 0 671173123 4096 1260 0 0 0 17 5 0 0 38 0 0 0 0 0 0 0 0 0 0
panyue@Saltedfish  /proc/1  cat status
Name:   systemd
Umask:  0000
State:  S (sleeping)
Tgid:   1
Ngid:   0
Pid:    1
PPid:   0
TracerPid:      0
Uid:    0       0       0       0
Gid:    0       0       0       0
FDSize: 256
Groups:
NStgid: 1
NSpid:  1
NSpgid: 1
NSsid:  1
VmPeak:   299988 kB
VmSize:   234452 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:      8844 kB
VmRSS:      8844 kB
```

图 4.2 stat 和 status 文件内容

这里为了方便读取，针对 stat 进行操作，我们需要的以下几位的数据：

1）第 1 位：pid，进程号

2）第 2 位：name，进程名

3）第 3 位：status，进程状态

4）第 4 位：ppid，父进程号

5）第 18 位：priority，优先级

6）第 23 位：memory，占用内存

②CPU 详细信息

CPU 相关的信息被存放在/proc/cpuinfo 文件里，按行列出了每个 CPU 每个属性和其对应的值。

```
panyue@Saltedfish  /proc  cat cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
stepping        : 9
microcode       : 0x84
cpu MHz         : 910.256
cache size      : 6144 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dt
s acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon peb
s bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monit
or ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadl
ine_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ibr
s ibpb stibp tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invp
cid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts h
wp_hwp_notify hwp_act_window hwp_epp
```

图 4.3 cpuinfo 文件内容

我们需要的是以下几行的信息：

1）第 5 行：model name，CPU 名

2）第 8 行：cpu MHz，CPU 主频

3）第 9 行：cache size，缓存大小

4）第 13 行：cpu cores，CPU 核数

5）第 25 行：address sizes，地址空间

此外，CPU 的利用率还需要由计算获得，其数据在/proc/stat 文件内，该文件每行记录了一个 CPU 的数据，第 1 位为 CPU 名，第 2 到 5 位分别为 user, nice, system 和 idle，则有：

cpu(total) = user + nice + system + idle

utilization = 100% * (△total － △idle) / △total

③内存信息

内存相关的信息被存放在/proc/meminfo 文件里，也按行列出了内存的每个属性和其对应的值。

```
panyue@Saltedfish  /proc  cat meminfo
MemTotal:       16346980 kB
MemFree:        11652476 kB
MemAvailable:   13511184 kB
Buffers:          614156 kB
Cached:          1250300 kB
SwapCached:            0 kB
Active:          3210084 kB
Inactive:         749652 kB
Active(anon):    2097732 kB
Inactive(anon):    19032 kB
Active(file):    1112352 kB
Inactive(file):   730620 kB
Unevictable:         424 kB
Mlocked:             424 kB
SwapTotal:       8388604 kB
SwapFree:        8388604 kB
Dirty:              1324 kB
Writeback:             0 kB
AnonPages:       2095784 kB
Mapped:           696468 kB
Shmem:             21488 kB
Slab:             445312 kB
SReclaimable:     356684 kB
```

图 4.4 meminfo 文件内容

我们需要的是以下几行的信息：

1）第 1 行：MemTotal，总内存大小

2）第 2 行：MemFree，剩余可用内存大小

3）第 15 行：SwapTotal，交换空间大小

4）第 16 行：cpu cores，剩余可用交换空间大小

内存利用率直接利用(1 – free / total) * 100%即可。

④模块信息

模块相关的信息被存放在/proc/modules 文件里，按行列出了每个加载的模块的属性。

```
panyue@Saltedfish /proc cat modules
fuse 118784 3 - Live 0x0000000000000000
ccm 20480 6 - Live 0x0000000000000000
nls_iso8859_1 16384 1 - Live 0x0000000000000000
nls_cp437 20480 1 - Live 0x0000000000000000
vfat 24576 1 - Live 0x0000000000000000
fat 81920 1 vfat, Live 0x0000000000000000
btusb 53248 0 - Live 0x0000000000000000
btrtl 16384 1 btusb, Live 0x0000000000000000
btbcm 16384 1 btusb, Live 0x0000000000000000
btintel 24576 1 btusb, Live 0x0000000000000000
uvcvideo 110592 0 - Live 0x0000000000000000
bluetooth 638976 5 btusb,btrtl,btbcm,btintel, Live 0x0000000000000000
videobuf2_vmalloc 16384 1 uvcvideo, Live 0x0000000000000000
videobuf2_memops 16384 1 videobuf2_vmalloc, Live 0x0000000000000000
videobuf2_v4l2 28672 1 uvcvideo, Live 0x0000000000000000
videobuf2_common 53248 2 uvcvideo,videobuf2_v4l2, Live 0x0000000000000000
videodev 208896 3 uvcvideo,videobuf2_v4l2,videobuf2_common, Live 0x0000000000000000
joydev 24576 0 - Live 0x0000000000000000
media 45056 2 uvcvideo,videodev, Live 0x0000000000000000
mousedev 24576 0 - Live 0x0000000000000000
ecdh_generic 24576 1 bluetooth, Live 0x0000000000000000
snd_hda_codec_realtek 110592 1 - Live 0x0000000000000000
snd_hda_codec_generic 86016 1 snd_hda_codec_realtek, Live 0x0000000000000000
snd_hda_codec_hdmi 57344 2 - Live 0x0000000000000000
arc4 16384 2 - Live 0x0000000000000000
intel_rapl 24576 0 - Live 0x0000000000000000
x86_pkg_temp_thermal 16384 0 - Live 0x0000000000000000
```

图 4.5 modules 文件内容

每一行的信息含义如下：

1）第 1 位：modules name，模块名

2）第 2 位：used memory，占用内存

3）第 3 位：used times，使用次数

⑤系统信息

系统相关的信息分别可以在以下文件中获取：

主机名 Hostname: /etc/hostnames.

操作系统名 OS Name: /etc/issue.

操作系统类型（32 or 64）OS Type: sizeof(char *) * 8;

内核版本 Kernel Version: /proc/sys/kernel/osrelease.

GCC 版本 GCC Version: /proc/version.

开机时间 Uptime: /proc/uptime.

⑥网速与磁盘 I/O 速度

与网络相关的信息全部存放在/proc/net 目录下，其中关于发送和接收数据大小的信息在 dev 文件内：

```
panyue@Saltedfish /proc cat net/dev
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo colls carrier compressed
wlp3s0: 7545718   10140    0    0    0     0          0         0   810165    6228    0    0    0     0       0          0
    lo:   27931     283    0    0    0     0          0         0    27931     283    0    0    0     0       0          0
enp2s0:       0       0    0    0    0     0          0         0        0       0    0    0    0     0       0          0
panyue@Saltedfish /proc
```

图 4.6 net/dev 文件内容

该文件内每一行记录一个网卡的信息，我们只需要获取 Receive 的 bytes 和 Transmit 的 bytes 信息即可，记录每两秒前后的信息即可计算出上传和下载的速度。

与磁盘相关的信息存放在/proc/diskstats 中，我们需要第 6 位和第 10 位的信息，分别代表读的扇区数 rd_sectors 和写的扇区数 wr_sectors，记录每两秒前后的信息，然后利用以下公式即可计算出速率。

read speed = (Δrd_sectors/Δt) * (block_size / 1024)

write speed = (Δwr_sectors/Δt) * (block_size / 1024)

这里 block_size 是 512

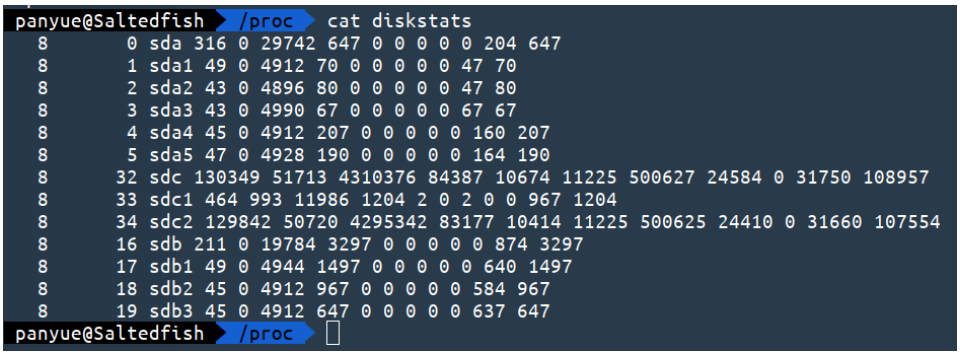

图 4.7 diskstats 文件内容

4.4 关键代码

（1）绘制曲线函数举例 – draw_cpu_curve

记录下一段时间内的 cpu 利用率在数组中，并逐个绘制曲线，每秒更新一次。

在绘图时让背景的网格线跟曲线一起向前动，使得曲线看上去更好看。

代码段 4.1 绘制 CPU 曲线

```
gboolean draw_cpu_curve(gpointer widget) {
    GtkWidget *cpu_curve = (GtkWidget *)widget;
    GdkColor color;
    GdkGC *gc = cpu_curve->style->fg_gc[GTK_WIDGET_STATE(widget)];
    static int flag = 0;
    static int now_pos = 0;
    int draw_pos = 0;

    /* Darw background */
    color.red = 0;
    color.green = 0;
    color.blue = 0;
    gdk_gc_set_rgb_fg_color(gc, &color);
    gdk_draw_rectangle(cpu_curve->window, gc, TRUE, 15, 30, 480, 200);

    /* Draw background lines */
```

```
    color.red = 0;
    color.green = 20000;
    color.blue = 0;
    gdk_gc_set_rgb_fg_color(gc, &color);
    for (int i = 30; i <= 220; i += 20)
        gdk_draw_line(cpu_curve->window, gc, 15, i, 495, i);
    for (int i = 15; i <= 480; i += 20)
        gdk_draw_line(cpu_curve->window, gc, i + cpu_curve_start, 30, i +
cpu_curve_start, 230);

    /* Settle cpu curve start position to make it live */
    cpu_curve_start -= 4;
    if (cpu_curve_start == 0)
        cpu_curve_start = 20;

    /* Initial data */
    if (flag == 0) {
        for (int i = 0; i < 120; i++) {
            cpu_ratio_data[i] = 0;
            flag = 1;
        }
    }

    /* Add data */
    cpu_ratio_data[now_pos] = cpu_ratio / 100;
    now_pos++;
    if (now_pos == 120)
        now_pos = 0;

    /* Draw lines */
    color.red = 0;
            color.green = 65535;
            color.blue = 0;
            gdk_gc_set_rgb_fg_color(gc, &color);
    draw_pos = now_pos;
    for (int i = 0; i < 119; i++) {
        gdk_draw_line(cpu_curve->window, gc,
                              15 + i * 4, 230 - 200 * cpu_ratio_data[draw_pos % 120],
                              15 + (i + 1) * 4, 230 - 200 * cpu_ratio_data[(draw_pos +
1) % 120]);
        draw_pos++;
        if (draw_pos == 120)
```

```
        draw_pos = 0;
    }

    /* Reset the color */
    color.red = 25000;
            color.green = 25000;
            color.blue = 25000;
            gdk_gc_set_rgb_fg_color(gc, &color);

    /* To loop this function, it must return TRUE */
    return TRUE;
}
```

（2）数据获取举例 – get_cpu_ratio

采用最普通的打开文件并读取的方式，并且系统维护一个全局变量用以存放上一次读取的值，用以绘图或者是计算速率等。

代码段 4.2 获取 CPU 信息

```
gboolean get_cpu_ratio(gpointer label) {
    /*
     * stat file format
     *
     * cpu user nice system idle iowait
     *
     * t1, t2: Two near moments
     * cpu(total) = user+nice+system+idle
     * pcpu = 100 *(total – idle) / total
     * total =total(t2) – total(t1)
     * idle =idle(t2) – idle(t1)
     */
    /* Data holder */
    static long old_idle, old_total;
    static int flag = 0;

    long user, nice, system, idle, total;
    float total_diff, idle_diff;
    char cpu[10], buffer[256], cpu_ratio_char[256];
    int fd;
    fd = open("/proc/stat", O_RDONLY);
    read(fd, buffer, sizeof(buffer));
    close(fd);
    sscanf(buffer, "%s %ld %ld %ld %ld", cpu, &user, &nice, &system, &idle);
```

```
    /* First */
    if (flag == 0) {
        flag = 1;
        old_idle = idle;
        old_total = user + nice + system + idle;
        cpu_ratio = 0;
    }
    /* Others */
    else {
        total = user + nice + system + idle;
        total_diff = total - old_total;
        idle_diff = idle - old_idle;
        cpu_ratio = 100 * (total_diff - idle_diff) / total_diff;
        total = old_total;
        idle = old_idle;
    }
    sprintf(cpu_ratio_char, "CPU usage: %0.1f%%", cpu_ratio);
    gtk_label_set_text(GTK_LABEL(label), cpu_ratio_char);
    return TRUE;
}
```

（3）解决编码问题 – utf8_fix

未经编码转换可能会导致程序崩溃。

代码段 4.3 编码转换

```
/*
 * utf8_fix - To settle waring: Invalid UTF-8 string passed to
pango_layout_set_text()
 *
 * Referencing from:
https://stackoverflow.com/questions/43753260/pango-warning-invalid-utf-8-string
-passed-to-pango-layout-set-text-in-gtk
 */
char* utf8_fix(char *c) {
    return g_locale_to_utf8(c, -1, NULL, NULL, NULL);
}
```

（4）解决自动跳转问题 – scroll_to_line

这个功能用于搜索进程信息，刷新后保证进度条在原位置。

代码段 4.4 进度条位置跳转

```
void scroll_to_line(gpointer scrolled_window, gint line_num, gint to_line_index) {
    GtkAdjustment *adj;
    gdouble lower_value, upper_value, page_size, max_value, line_height,
to_value;
    adj =
gtk_scrolled_window_get_vadjustment(GTK_SCROLLED_WINDOW(scrolled_w
indow));
    lower_value = gtk_adjustment_get_lower(adj);
    upper_value = gtk_adjustment_get_upper(adj);
    page_size = gtk_adjustment_get_page_size(adj);
    max_value = upper_value - page_size;
    line_height = upper_value / line_num;
    to_value = line_height * to_line_index;
    if (to_value < lower_value)
        to_value = lower_value;
    if (to_value > max_value)
        to_value = max_value;
    gtk_adjustment_set_value(adj, to_value);
    return;
}
```

（5）解决字体问题 – set_label_fontsize

该函数用于设置一个 label 中的字体，便于显示结果。

代码段 4.5 设置 label 中的字体

```
/*
 * set_label_fontsize - To set font size in a label
 *
 * Referencing from: https://blog.csdn.net/gl_ding/article/details/4939355
 */
void set_label_fontsize(GtkWidget *label, char *fontsize) {
    PangoFontDescription *desc_info =
pango_font_description_from_string(fontsize);
    gtk_widget_modify_font(label, desc_info);
    pango_font_description_free(desc_info);
}
```

4.5 调试记录与运行结果

在 build 文件夹下执行 make 后，可以看到可执行程序 py_sysmonitor，为测系统监视器的可执行程序。

运行程序的截图如下，可以看到系统监视器正常运行，效果良好。

图 4.8 进程管理页面

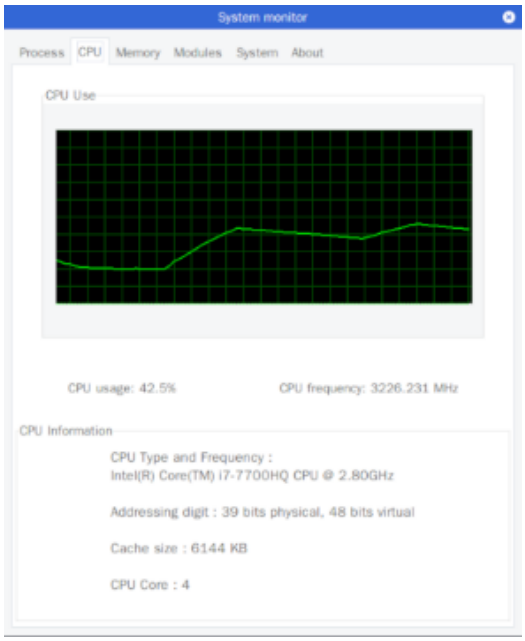

图 4.9 CPU 信息页面

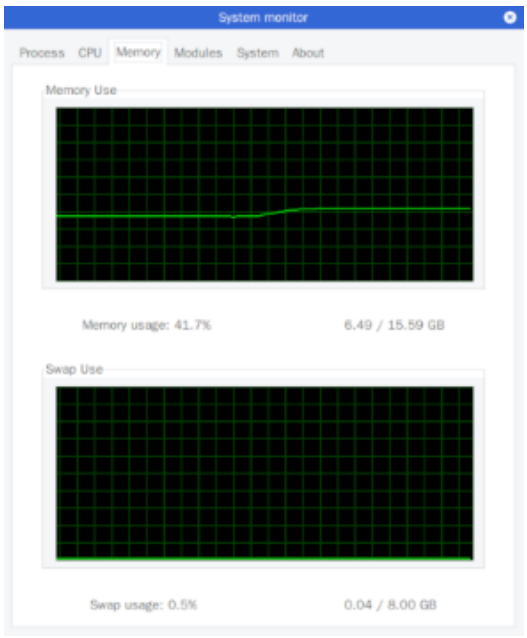

图 4.10 内存信息页面

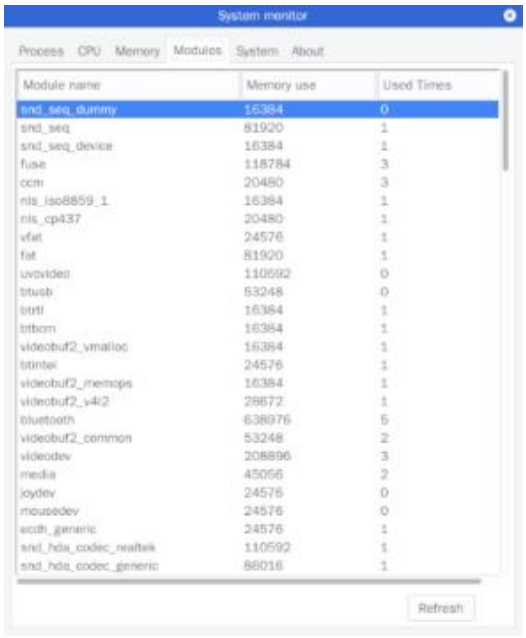

图 4.11 模块信息页面

图 4.12 内存信息页面

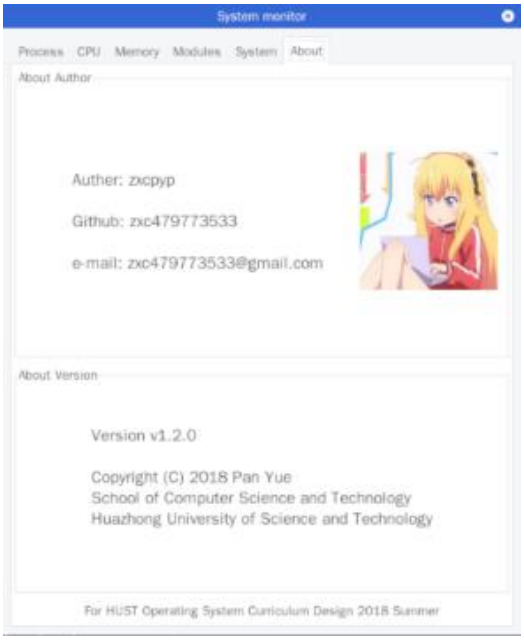

图 4.13 模块信息页面

设计内容五 模拟文件系统

5.1 实验目的

（1）掌握实例操作系统的实现方法。

（2）设计并实现一个模拟的文件系统。

5.2 设计内容

（1）基于一个大文件(如 100M)，模拟磁盘。

（2）格式化，建立文件系统管理数据结构。

（3）实现文件/目录创建/删除，目录显示等基本功能(可自行扩充文件读/写、用户登录、权限控制、读写保护等其他功能)。

5.3 实验环境及步骤

5.3.1 实验环境

本次课程设计使用的环境配置如下：

（1）操作系统版本：Arch Linux x86_64

（2）操作系统内核版本：4.17.8

（3）编译器及其版本：GCC version 8.1.1

（4）自动编译工具：CMake version 3.11.4

（5）编程环境：Visual Studio Code

5.3.2 文件卷设计

本课程设计分两部分，一是模拟磁盘格式化，建立数据结构，同时要维护所有的结构和变量。二是设计一个框架来对格式化的磁盘进行操作，这包括实现操作命令以及调用框架。

本次课程设计模拟一块 64MB 大小的磁盘，磁盘每一块的大小为 1KB。第 0 块存放用户，第一块存放超级块，接着划分 1024 个 inode 的空间，最大允许 i 结点数 1024，后面跟着其他的 1KB 普通块。

从第 0 块到 1026 块大小即为数据结构的大小，数据紧密排列。从 1027 开始的普通块按每块 1KB 的大小排列。

设计三个全局变量维护超级块，i 结点，空闲块开始的位置，利用 fseek 函数改变文件偏移量的方法去读/写对应的块。

整个文件卷的设计如下图：

图 5.1 文件卷设计

5.3.3 磁盘数据结构设计

（1）超级块 – super_block

超级块的结构如下：

```
typedef struct super_block {
   int inode_map[INODENUM];
   int block_map[BLOCKNUM];
   int inode_free_num;
   int block_free_num;
} super_block;
```

采用了位示图的方式来管理 i 结点和空闲块。位示图是利用二进制的一位来表示磁盘中的一个盘块的使用情况。当其值为“0”时，表示对应的盘块空闲；为“1”时，表示已经分配。磁盘上的所有盘块都有一个二进制位与之对应，这样，由所有盘块所对应的位构成一个集合，称为位示图。

此外超级块维护两个变量，表示空间 i 结点和空闲块数目。

（2）i 结点 – inode

i 结点的结构如下：

```
typedef struct inode {
    int block_used[FILEBLKMAX];
    int block_used_num;
    int size;
    int mode;
    time_t creat_time;
    time_t modify_time;
    int user_id;
} inode;
```

i 结点的设计仿照了标准 Linux 的设计，存放了该文件占用的块号，占用块数目，文件大小，文件权限，创建时间，最后修改时间，所属用户 id。其中判断一个文件是普通文件还是目录要利用权限 mode 的最高位。

（3）目录项 – directory

目录项的结构如下：

```
typedef struct directory {
    char name[FILENAMEMAX];
    int inode_id;
} directory;
```

目录项即目录文件内存放的数据，其要么指向下一级目录，要么指向一个普通文件，这个“指向”利用 i 结点编号实现。

整个磁盘的数据结构大致如下图：

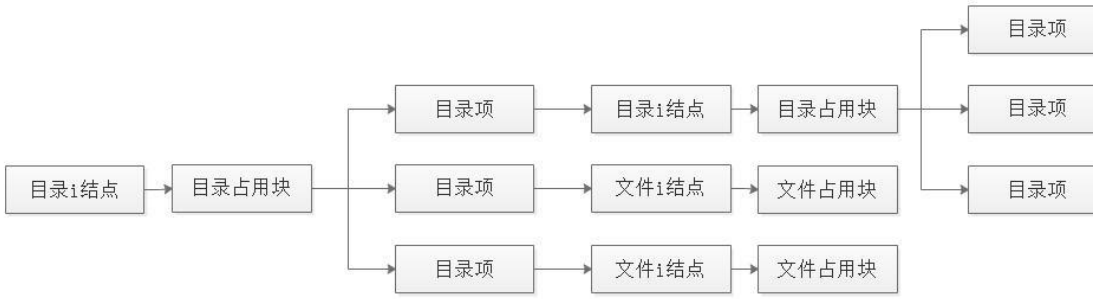

图 5.2 磁盘数据结构设计

5.3.4 核心函数设计

核心的函数即块管理的四个基本函数：inode_alloc，inode_free，block_alloc，block_free，以及两种 i 结点初始化函数 init_dir_inode 和 init_file_inode。有了这六个函数，我们就可以实现其他所有的功能。

（1）alloc 函数设计

两个 alloc 函数的算法是一致的，即：

①检查是否还有剩余块，若没有，返回报错没有空闲块；

②扫描位示图，寻找第一个空闲标志位为 0 的块；

③将该块空闲标记置为 1，空闲块数目减 1，返回该块块号。

（2）block_free 函数设计

空闲块的 free 很简单，只需要回收块号即可，即：

①将该块空闲标记置为 0，空闲块数目加 1.

（3）inode_free 函数设计

i 结点的 free 不仅要回收 i 结点本身占用的块，还要回收其管理的所有空闲块，即：

①根据 fseek 函数读取 i 结点的信息；

②对该 i 结点管理的所有块执行 block_free 函数；

③将该块空闲标记置为 0，空闲块数目加 1.

这里的 alloc 和 free 均不涉及磁盘上的数据更改，仅仅是块号的分配和回收，这样便于我们操作，而且空闲块上有数据也不怕，因为它会在下次分配时被清除掉。

（4）结点初始化函数设计

这个函数的操作很简单，设置好 i 结点结构体的初始值即可，目录和文件的区别在于对于目录，在创建之时就应该含有初始的两个目录项，分别是“.”和“..”，即指向自身和上一级目录的目录项。

这六个核心函数的代码见 5.4 章节 – 关键代码中。

5.3.5 多用户和权限设计

（1）数据结构

多用户的信息存储在磁盘第 0 块上，利用如下两个结构体：

①用户信息

```
typedef struct user {
   char user_name[USERNAMEMAX];
   char user_pwd[USERPWDMAX];
} user;
```

②全体用户

```
typedef struct sys_users {
   int user_map[USERNUMMAX];
   int user_num;
   user users[USERNUMMAX];
} sys_users;
```

在登录用户时，在这些数据中逐一匹配用户名和密码，在登录之后登记全局变量当前用户 id 即可标识用户。、

（2）权限管理

权限设计也仿照标准 Linux 设计了 drwxrwxrwx 的权限，第 1 位 d 代表文件，后面的 r 代表可读，w 代表可写，x 代表可执行。第一组代表自己对该文件的权限，第二组代表同一用户组中其他用户对该文件的权限，第三组代表其他用户对该文件的权限。

实际上，由于是简化的文件系统，相比 OS，这里只用到了第一位的文件属性判断和第一组，第三组的 rw 位进行读写权限判断。

权限的存储采用十进制数字，但为了便于权限的判断，设计了一个 8 进制转 10 进制的函数，就可以方便的利用 8 进制数管理权限。

（3）权限保护

本次课程设计中的权限设计如下：

磁盘中，以下几个操作需要读权限支持：

进入目录，列出当前目录下的文件及其详细信息，读取文件内的信息。

以下几个操作需要写权限支持：

创建文件（包括创建目录，普通文件，复制和移动），删除文件，保存文件数据。

以下几个操作需要所属用户权限支持：

修改权限，修改密码。

以下几个操作需要 root 权限支持：

添加、删除用户，格式化磁盘。

5.3.6 操作命令设计

文件系统共为磁盘提供了以下几种命令：

表 5.1 磁盘操作命令设计

编号	命令	参数	功能	命令类型
1	mkdir	<目录名>	创建目录	目录和文件操作
2	rmdir	<目录名>	删除目录	目录和文件操作
3	cd	<目录名>	进入目录	目录和文件操作
4	ls [-l]	无	列出当前目录下文件信息 [-l	目录和文件操作

			为列出详细信息]	
5	touch	<文件名>	创建空文件/更新文件最后修改时间	目录和文件操作
6	vim	<文件名>	编辑文件内容	目录和文件操作
7	cat	<文件名>	输出文件内容	目录和文件操作
8	cp	<源文件> <目标文件/目录>	复制文件	目录和文件操作
9	mv	<源文件> <目标文件/目录>	移动文件	目录和文件操作
10	useradd	<用户名> <密码>	添加用户	用户管理操作
11	userdel	<删除用户>	删除用户	用户管理操作
12	passwd	无	修改密码	用户管理操作
13	fmt	无	格式化磁盘	其他操作
14	chmod	<权限值> <文件/目录>	修改权限	其他操作
15	help	无	打印帮助信息	Shell 内建操作
16	exit	无	退出文件系统	Shell 内建操作
17	reset	无	初始化磁盘	开发者调试操作
18	puid	无	打印当前用户 id	开发者调试操作
19	pino	无	打印当前目录 i 结点号	开发者调试操作
20	dirnum	无	打印当前目录下目录项数目	开发者调试操作
21	show	无	打印当前目录下目录项信息	开发者调试操作
22	users	无	打印磁盘中所用用户信息	开发者调试操作
23	superi	<序号>	打印超级块中记录的某一 i 结点块使用状态	开发者调试操作
24	superb	<序号>	打印超级块中记录的某一普通块使用状态	开发者调试操作

目录和文件操作的设计：

（1）mkdir、touch – 创建目录、创建空文件

这两个函数的算法类似，而且在编程时使用了同一个接口。

①可行性检查，包括权限，重名，剩余空间，对于 touch，若文件存在则更新最后修改时间，退出函数；

②判断是否需要增加一块（对目录文件来说），若是，则调用 block_alloc 分配一块；

③分配新的 i 结点块；

④根据类型是目录还是普通文件，调用初始化函数初始化 i 结点；

⑤在目录包含的目录项中注册该目录项。

（2）rmdir、rm – 删除目录、删除文件

这两个函数的算法也类似，同样适用了同一个接口

① 可行性检查，包括权限，重名，剩余空间，类型，删除目录的话，还不能删除“.”和“..”以及非空文件；

②回收 i 结点块；

③如果当前目录下出现一块完整的空块，将其回收。

（3）cd – 进入目录

①可行性检查，包括是否存在，是否是目录，权限；

②打开新的目录并更新全局变量；

③修改路径显示。

（4）ls 与 ls-1 – 列出详细信息

①可行性检查，包括权限；

②循环获取每个目录项 i 结点的信息并打印。

（5）vim – 编辑文件、

vim 的实现调用了外部接口，并采用在/tmp 目录下生成一个缓冲文件的方法，打开文件即是从磁盘块上读数据到缓冲文件，接着 vim 打开缓冲文件并编辑。保存文件即是把缓冲文件上的数据写回磁盘。

①可行性检查，包括权限，是否存在，是否是普通文件，是否只读；

②将磁盘块上数据读到缓冲文件

③使用 vim 打开缓冲文件，并编辑

④检查是否可写，不可写则将文件还原并提升，否则将新的数据写回磁盘。

（6）cat – 输出文件内容

①可行性检查，包括权限，是否存在，是否是普通文件；

②输出文件内容。

（7）cp – 文件的复制

①可行性检查，包括权限，是否存在，是否是普通文件（这里没有实现递归复制目录的功能）；

②将文件内容读到缓冲文件中；

③判断是复制到其他目录内还是当前目内，若是其他目录，跳至⑥；

④若是当前目录，则执行创建新文件操作；

⑤将缓冲文件中的内容写到新文件里。

⑥若是其他目录，则进入其他目录；

⑦可行性检查，包括权限，是否重名；

⑧执行创建新文件操作；

⑨将缓冲文件中的内容写到新文件里。

（8）mv – 文件的移动

mv 相比 cp 就要简单很多，不需要复制文件和 i 结点，只需要在目录项层次上操作即可。

①可行性检查，包括权限，是否存在，是否是普通文件；

②判断是移动到其他目录内还是当前目内，若是其他目录，跳至⑤；

③若是当前目录，则创建新的目录项指向源文件的 i 结点；

④删除原目录项；

⑤若是其他目录，则进入其他目录；

⑥可行性检查，包括权限，是否重名；

⑦创建新的目录项指向源文件的 i 结点；

⑧删除原目录项。

目录操作单纯的是对第 0 块上结构体数据的操作，比较简单，这里不再赘述，代码详见附录。

此外，还有两个十分重要的操作：

（1）fmt – 磁盘格式化

磁盘格式化是建立磁盘的第一部，是一切命令的基础。

①将超级块设置为初始状态，并写回磁盘；

②将 root 目录的 i 结点设置为初始状态，并写回磁盘；

③在 root 目录下创建“.”和“..”两个目录项；

④将路径设置为初始状态“/”；

格式化不需要去清空原有的各种数据，只要把管理全局超级块处理掉，就实现了功能。

（2）reset

完成上面所有的功能后，我们并不能进入磁盘，因为没有任何一个用户，于是在此设计了一个开发者调试操作，用来把磁盘初始化，设置初始 root 用户以及初始密码。

①清空用户结构；

②添加第 0 个用户 root，设置初始密码为 123456

③格式化磁盘。

5.3.7 执行框架设计

执行框架的设计是在我做了 CMU 的 shell lab 后受到启发，模仿设计的类 s

hell 框架，包含命令行解析，内建命令和外部命令判断与执行，执行截图如下：

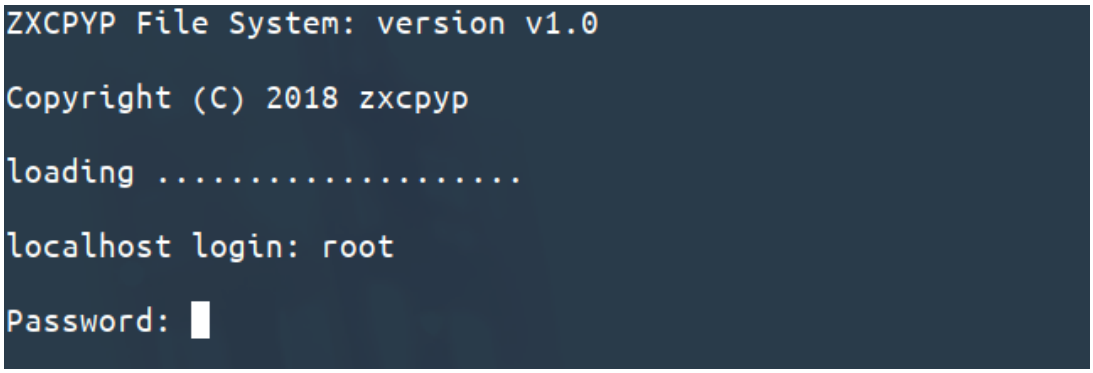

图 5.3 登录界面截图

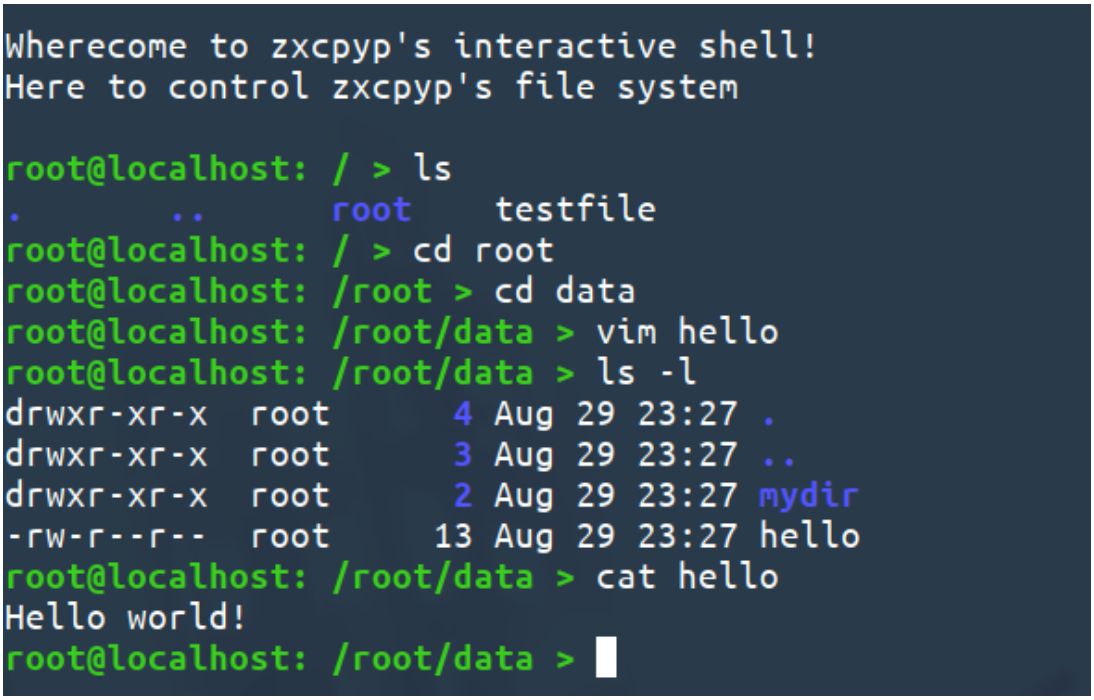

图 5.3 shell 界面截图

5.4 关键代码

首先是上文提到的核心函数，由他们作为基础来构建其他的命令操作。

（1）i 结点分配 – inode_alloc

代码段 5.1 i 结点分配

```
int inode_alloc(void) {
    int ino;
    if (super.inode_free_num <= 0)
        return FS_NO_INODE;
    super.inode_free_num--;
    for (ino = 0; ino < INODENUM; ino++) {
        if (super.inode_map[ino] == 0) {
```

```
            super.inode_map[ino] = 1;
            break;
        }
    }
    return ino;
}
```

（2）i 结点释放 – inode_free

代码段 5.2 i 结点释放

```
int inode_free(int ino) {
    inode node;
    fseek(disk, INODEPOS + ino * INODESIZE, SEEK_SET);
    fread(&node, sizeof(inode), 1, disk);
    for (int i = 0; i < node.block_used_num; i++)
        block_free(node.block_used[i]);
    super.inode_map[ino] = 0;
    super.inode_free_num++;
    return FS_OK;
}
```

（3）普通块分配 – block_alloc

代码段 5.3 普通块分配

```
int block_alloc(void) {
    int bno;
    if (super.block_free_num <= 0)
        return FS_NO_BLOCK;
    super.block_free_num--;
    for (bno = 0; bno < BLOCKNUM; bno++) {
        if (super.block_map[bno] == 0) {
            super.block_map[bno] = 1;
            break;
        }
    }
    return bno;
}
```

（4）普通块释放 – block_free

代码段 5.4 普通块释放

```
int block_free(int bno) {
```

```
    super.block_free_num++;
    super.block_map[bno] = 0;
    return FS_OK;
}
```

（5）目录结点初始化 – init_dir_inode

代码段 5.5 目录结点初始化

```
int init_dir_inode(int new_ino, int ino) {
    int bno;
    inode node;
    directory basic_link[2];

    fseek(disk, INODEPOS + new_ino * INODESIZE, SEEK_SET);
    fread(&node, sizeof(inode), 1, disk);
    bno = block_alloc();

    /* Set new inode information */
    node.block_used[0] = bno;
    node.block_used_num = 1;
    node.size = 2 * sizeof(directory);
    node.mode = oct2dec(1755);
    time_t timer;
            time(&timer);
    node.creat_time = timer;
    node.modify_time = timer;
    node.user_id = current_user_id;

    /* Save inode information */
    fseek(disk, INODEPOS + new_ino * INODESIZE, SEEK_SET);
    fwrite(&node, sizeof(inode), 1, disk);

    /* Set basic links */
    strcpy(basic_link[0].name, ".");
    basic_link[0].inode_id = new_ino;
    strcpy(basic_link[1].name, "..");
    basic_link[1].inode_id = ino;

    /* Save basic links */
    fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
    fwrite(basic_link, sizeof(directory), 2, disk);
```

```
    return FS_OK;
}
```

（6）文件结点初始化 – init_file_inode

代码段 5.6 文件结点初始化

```
int init_file_inode(int new_ino) {
    inode node;
    fseek(disk, INODEPOS + new_ino * INODESIZE, SEEK_SET);
    fread(&node, sizeof(inode), 1, disk);

    /* Set new inode information */
    node.block_used_num = 0;
    node.size = 0;
    node.mode = oct2dec(644);
    time_t timer;
            time(&timer);
    node.creat_time = timer;
    node.modify_time = timer;
    node.user_id = current_user_id;

    /* Save inode information */
    fseek(disk, INODEPOS + new_ino * INODESIZE, SEEK_SET);
    fwrite(&node, sizeof(inode), 1, disk);

    return FS_OK;
}
```

然后是基本的磁盘格式化操作：

（7）磁盘格式化 – disk_format

代码段 5.3 普通块分配

```
int format_disk(void) {
    int ret;

    /* Set inode map */
    memset(super.inode_map, 0, sizeof(super.inode_map));
    super.inode_map[0] = 1;
    super.inode_free_num = INODENUM - 1;

    /* Set block map */
```

```
    memset(super.block_map, 0, sizeof(super.block_map));
    super.block_map[0] = 1;
    super.block_free_num = BLOCKNUM - 1;

    /* Set root inode */
    current_inode_id = 0;
    fseek(disk, INODEPOS, SEEK_SET);
    ret = fread(&current_inode, sizeof(inode), 1, disk);
    if (ret != 1)
        return FS_RD_ERROR;
    current_inode.block_used[0] = 0;
    current_inode.block_used_num = 1;
    current_inode.size = 2 * sizeof(directory);
    current_inode.mode = oct2dec(1755);
    time_t timer;
            time(&timer);
    current_inode.creat_time = timer;
    current_inode.modify_time = timer;
    current_inode.user_id = 0;

    /* Set basic link of root file */
    current_dir_num = 2;
    strcpy(current_dir_content[0].name, ".");
    current_dir_content[0].inode_id = 0;
    strcpy(current_dir_content[1].name, "..");
    current_dir_content[1].inode_id = 0;

    strcpy(path, "root@localhost: / >");

    return FS_OK;
}
```

5.5 调试记录与运行结果

在 build 文件夹下执行 make 后，可以看到可执行程序 zxcpypfs，为磁盘配套的 shell 的可执行程序。

执行./zxcpypfs –h，可以看到如下图的提示：

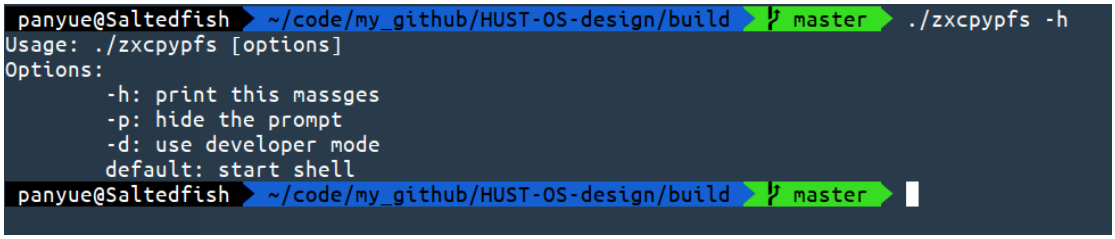

图 5.4 提示信息

在可执行程序后加 –p，代表不打印命令提示符，加-d，代表进入开发者模式，可以使用开发者命令。

本次测试的内容如下表：

表 5.2 文件系统测试内容

项目	测试	命令	测试结果
1	测试创建目录	mkdir test	通过
2	测试创建文件	touch data	通过
3	测试列出内容	ls -l	通过
4	测试向文件写数据	vim data	通过
5	测试从文件读数据	cat data	通过
6	测试移动文件	mv data testmv	通过
7	测试复制文件	cp testmv /test	通过
8	测试只读权限	chmod 444 testmv	通过
9	测试只写权限	chmod 244 testmv	通过
10	测试添加用户	useradd zxcpyp 888888	通过
11	测试删除用户	userdel zxcpyp	通过
12	测试格式化	fmt	通过

（1）测试项目 1&2&3

命令的执行和结果如下图，ls 的结果显示文件是空文件。

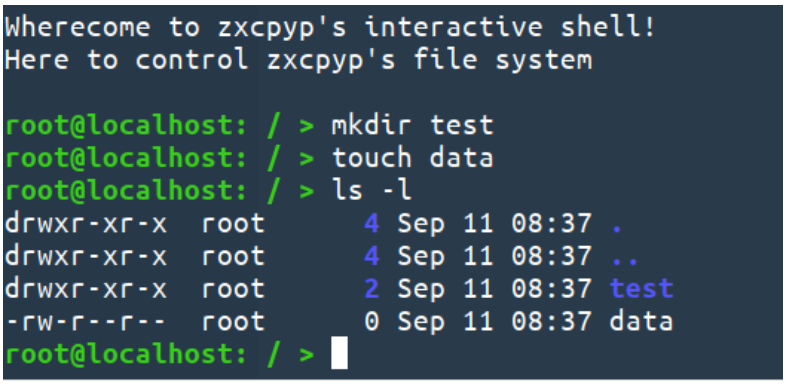

图 5.5 测试项目 1&2&3 创建空目录和空文件

（2）测试项目 4&5

首先执行 vim，写入数据：

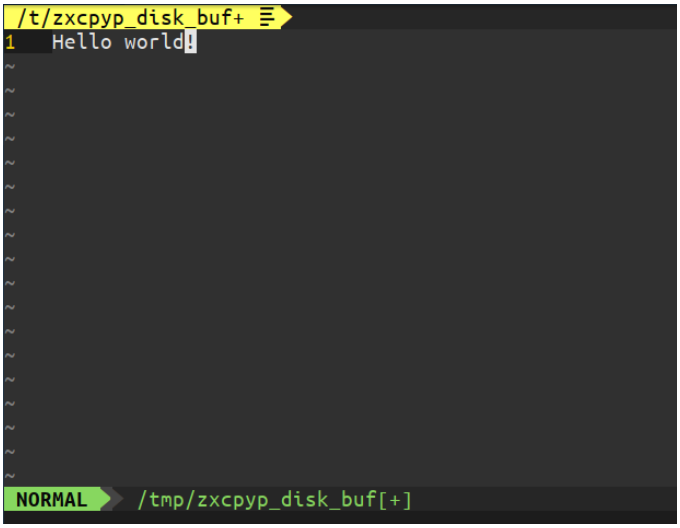

图 5.6 测试项目 4&5 vim 写入数据

接着保存并写入文件，如下图，可以看到输出了文件内容 Hello world!

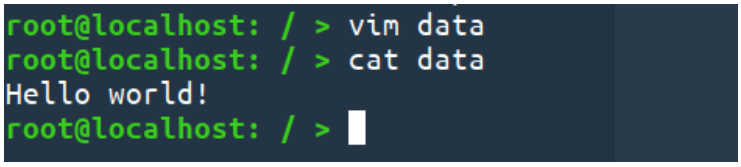

图 5.7 测试项目 4&5 保存，并输出测试

（3）测试项目 6

命令和执行的结果如下图，可以看到此时 testmv 大小已经变成了 13.

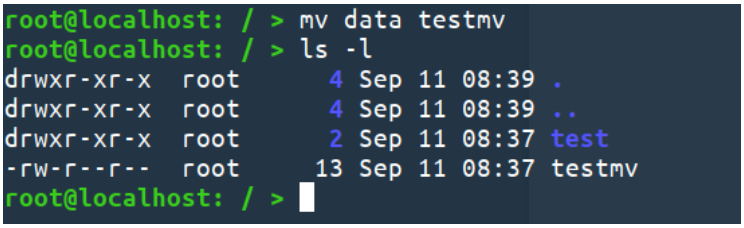

图 5.8 测试项目 6 移动文件并查看属性

（4）测试项目 7

命令的执行和结果如下图，将文件复制到了 test 目录下。

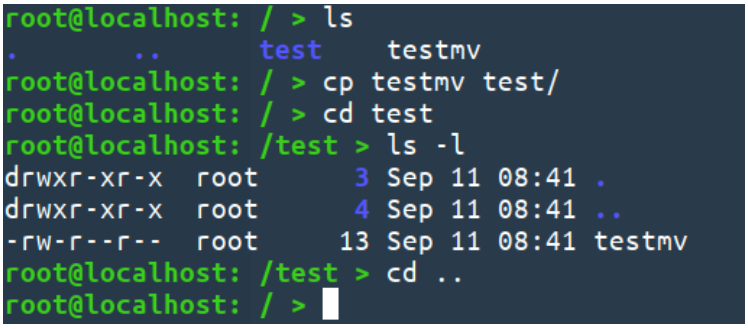

图 5.9 测试项目 7 复制文件并查看属性

（5）测试项目 8

更改 testmv 文件的权限为只读，执行 cat 和 vim 测试，发现 test 可以输出信息，但 vim 可读不可写。

```
root@localhost: / > ls -l
drwxr-xr-x  root      4 Sep 11 08:41 .
drwxr-xr-x  root      4 Sep 11 08:41 ..
drwxr-xr-x  root      3 Sep 11 08:41 test
-rw-r--r--  root     13 Sep 11 08:37 testmv
root@localhost: / > chmod 444 testmv
root@localhost: / > ls -l
drwxr-xr-x  root      4 Sep 11 08:42 .
drwxr-xr-x  root      4 Sep 11 08:42 ..
drwxr-xr-x  root      3 Sep 11 08:41 test
-r--r--r--  root     13 Sep 11 08:37 testmv
root@localhost: / > cat testmv
Hello world!
root@localhost: / > vim testmv
vim: Fail to save "testmv": Insufficient privilege
root@localhost: / >
```

图 5.10 测试项目 8 更改文件权限为只读并测试

（6）测试项目 9

更改 testmv 文件权限为只写，发现无法使用 cat 和 vim 获取文件内容。

```
root@localhost: / > ls -l
drwxr-xr-x  root      4 Sep 11 08:44 .
drwxr-xr-x  root      4 Sep 11 08:44 ..
drwxr-xr-x  root      3 Sep 11 08:41 test
-r--r--r--  root     13 Sep 11 08:37 testmv
root@localhost: / > chmod 244 testmv
root@localhost: / > ls -l
drwxr-xr-x  root      4 Sep 11 08:44 .
drwxr-xr-x  root      4 Sep 11 08:44 ..
drwxr-xr-x  root      3 Sep 11 08:41 test
--w-r--r--  root     13 Sep 11 08:37 testmv
root@localhost: / > cat testmv
cat: "testmv": Insufficient privilege
root@localhost: / > vim testmv
vim: Fail to open "testmv": Insufficient privilege
root@localhost: / >
```

图 5.11 测试项目 9 更改文件权限为只写并测试

（7）测试项目 10&11

执行命令添加用户，并使用新用户登录。

```
root@localhost: / > useradd zxcpyp 888888
root@localhost: / > exit
[EXIT] User-exit. Terminated!
 panyue@Saltedfish  ~/code/my_github/HUST-OS-design/build  master  ./zxcpypfs
ZXCPYP File System: version v1.2

Copyright (C) 2018 zxcpyp

loading ....................

localhost login: zxcpyp

Password:
```

图 5.12 测试项目 10&11 添加新用户

使用开发者模式以新用户身份进入系统，可以看到用户名已经更新了，执行开发者用命令 users，可以看到当前系统中存在的所有用户。

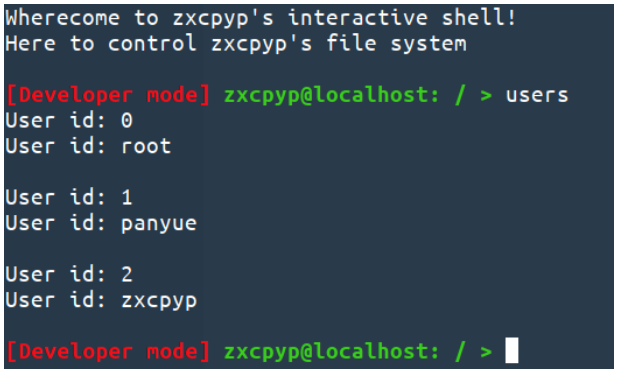

图 5.13 测试项目 10&11 查看所有用户

接着换用 root 身份进入系统，执行删除用户命令，再使用 users 查看发现刚才添加的用户已经被删掉了。

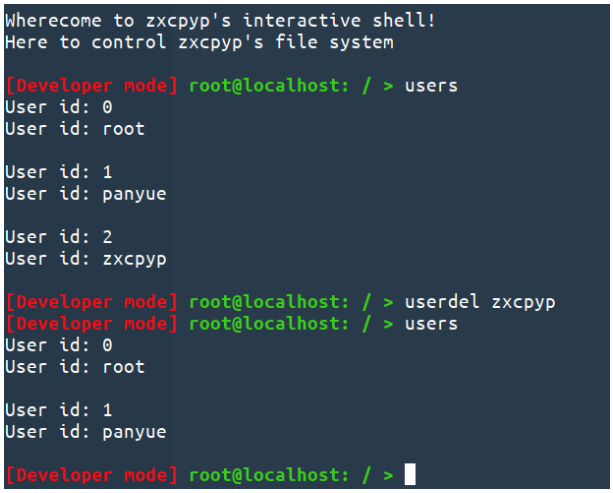

图 5.14 测试项目 10&11 查看所有用户

（8）测试项目 12

在文件系统中建立些许文件，执行 fmt 命令，可以看到系统已经恢复了初始状态，完成了格式化。

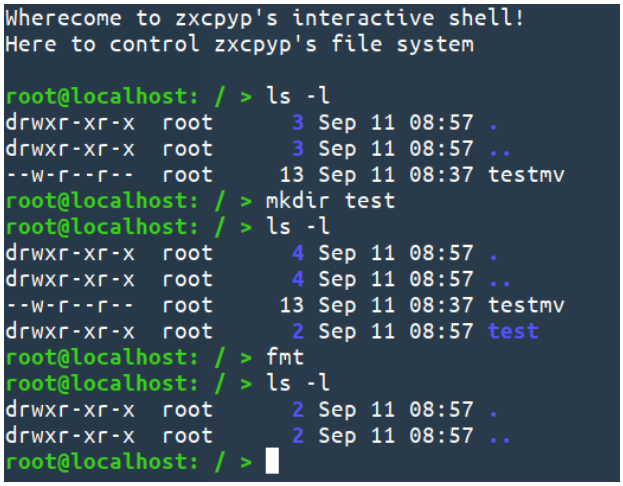

图 5.15 测试项目 12 执行格式化

通过以上 12 个测试，基本展现了文件系统的各个方面，也验证了系统的正确性。

实验感想与收获

本次课程设计是继 C 语言、数据结构、数据库系统之后的又一个课程设计，也是我到目前为止觉得难度最大的一个课程设计，整个课设大概断断续续花费了接近两周的时间，代码一共写了 7000 多行（除去 sys.c 的 2000 行），并且花了大量的时间和精力在调试上，但完成之后也是我收获最多的一个课设了。

实验的环境没有选择文档里推荐的 Ubuntu 系统 14.04，事实上我以前也在这个系统下进行过操作，只需要很简单几部就可以完成。但出于自己的喜好，采用了自己平常使用的 Arch Linux 以及最新版内核 4.17.8，同时也因为使用了最新版内核，导致网上没有任何中文资源，在实验的时候踩了无数坑。

编译完内核后，Arch 上并没有 Ubuntu 上那样的 make install 命令，需要自行拷贝内核镜像，构建启动盘。同时系统调用的添加方法和以前的完全不一样了。通过在 StackOverflow 中查阅了解到在 Arch 中已经重写了所有的系统调用，利用了很多宏来构建系统调用，所以只能仿照着现有的系统调用源码去实现自己的系统调用。在寻找新版本内核中出现的问题中花费了很长的时间，但总是收获还是很多了，通过这个过程了解了新版本内核的结构，看了很多源码，也对内核做了很多调试。同时也在做这一设计时阅读了 Robert Love 的 Linux 内核设计与实现，尽管书是用 2.6 版本的老内核编写的，但还是让我明白了很多内核编程和设计的思想。坑踩的越多，学到的也越多。

设备驱动相对来说就很简单了，这次实现了自动生成和删除设备文件的功能，也是通过网上查阅资料学到了，通过这个实验也让我熟悉了设备驱动的构成和接口函数，如果以后有机会的话回去读一读 Linux 设备驱动这本书。

而选做题的系统监控器和文件系统就有点复杂了。这算是我第一次正经的使用图形库开发带有图形界面的程序，以前一直没用过，正好利用这个机会系统看教程学习了 GTK 编程，掌握了一个图形库的用法。至于系统监视器其他的部分，就是大量的工程去读取展示数据了，本身并不难，但需要获取的东西特别多，但都在/proc 目录下。

而文件系统就需要大量的设计工作了。起初我实验的是链接文件来管理块，而在我写了两天后发现 bug 越来越多，最后不得已重构了代码重新设计。为此我专门花了半天的时间，在几张草稿纸上画好了文件卷分区，目录结构设计，相关操作的算法流程，重新选用位示图作为管理空闲块的办法。在核对几遍算法之后再一口气的写完文件系统库，整个流程就舒服了很多。

作为文件系统的调用，由于之前写过 CMU 的 shell lab，直接魔改框架做了一个简易的 shell 来调用命令，创建一个开发者模式也是在后期想到的注意。既然调试的时候到处打 log，不用了又删掉或者注释掉，那为什么不专门设计一组用来调试的接口呢？于是又做了好几个用来获取系统内部结构体数据的函数，用以调试，也加快了整个系统调试的进度。

剩下的操作就是想着怎么去让我的文件系统尽可能像 Linux，为此查阅了 printf 的颜色代码，使得可以输出彩色提示符，并且区分目录与普通文件，同时又针对我设计的结构重新编写了 ls –l 程序，通过外部缓冲区和调用 Linux 的 vim 实现了文件的编辑，实现了路径替换等等，总之在文件系统上花的时间还是不少的。

总结一下，这次课程设计对我来说，让我收获最大的还是选择了新版的内核，毕竟以前有过基础，很熟悉旧内核的各种操作，这次的不断踩坑促使我不断地去 google，在各种英文网站上搜寻最新的解答，促使我去阅读内核源码，从而了解很多内核的结构，这个收获可以说是很满意了，课设的基本目标也早实现了。文件系统的设计也让我更进一步理解了文件系统的块管理，文件目录的实现。最后实现了一个完整的文件系统也挺令我满意的。

在做课程设计时也阅读了很多书本资料，像现代操作系统、Linux/UNIX 系统编程手册、Linux 内核设计与实现等等，总之，课设让人收获最大的并不是去完成任务写代码，而是为了去完成课设而自己搜索的，查阅的，参考的各种知识与资料。

附录 课程设计源代码

设计内容一 熟练掌握 Linux 编程环境

copy.c – 文件拷贝

```
/*
 * HUST OS Design - Part I
 *
 * Linux Basics - File copy
 *
 * Created by zxcpyp at 2018-07-17
 *
 * Github: zxc479773533
 */

#include "../lib/zxcpyp_sys.h"

#define BUF_LEN 1024

mode_t get_file_mode(char *file) {
    struct stat s_buf;
    mode_t file_mode = 0x0;
    if (stat(file, &s_buf) != 0)
        err_exit("Stat");
    return s_buf.st_mode;
}

void do_copy(int read_fd, int write_fd) {
    int read_num;
    char buf[BUF_LEN];
    for (;;) {
        read_num = read(read_fd, buf, BUF_LEN);
        if (read_num == -1)
            err_exit("Copy");
        else if (read_num == 0)
            break;
        write(write_fd, buf, read_num);
    }
}

int main(int argc, char **argv) {
    int read_fd, write_fd;
    /* Check args */
    if (argc != 3)
        usage_err("./copy <src> <dst>");
    /* Open files */
    if ((read_fd = open(argv[1], O_RDONLY)) == -1)
        err_exit("Open source file");
    if ((write_fd = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, get_file_mode(argv[1]))) == -1)
        err_exit("Open destnation file");
    do_copy(read_fd, write_fd);
    close(read_fd);
    close(write_fd);
    return 0;
}
```

fork_demo.c – 并发进程演示

```
/*
 * HUST OS Design - Part I
 *
 * Linux Basics - Fork demo using GTK+ 3.0
 *
 * Created by zxcpyp at 2018-07-17
 *
 * Github: zxc479773533
 */

#include "../lib/zxcpyp_sys.h"

#include <wait.h>
#include <gtk/gtk.h>

#define WINDOW_WIDTH 450
#define WINDOW_HEIGHT 300
#define RE_FEQ 1000

/*
 * refresh_time - Refresh the time in window1
 */
gboolean refresh_time(gpointer label) {
    time_t times;
    struct tm *time_buf;
    time(&times);
    time_buf = localtime(&times);

    /* Get now time */
    gchar *text_day = g_strdup_printf("<span font_desc='48'>%04d-%02d-%02d</span>", \
        1900 + time_buf->tm_year, 1 + time_buf->tm_mon, time_buf->tm_mday);
    gchar *text_time = g_strdup_printf("<span font_desc='32'>%02d:%02d:%02d</span>", \
        time_buf->tm_hour, time_buf->tm_min, time_buf->tm_sec);
    gchar *text_data = g_strdup_printf("\n%s\n\n%s\n", text_day, text_time);

    gtk_label_set_markup(GTK_LABEL(label), text_data);
    return TRUE;
}

/*
 * refresh_num - Refresh the num in window2
 */
gboolean refresh_num(gpointer label) {
    static int num = 0;
    gchar *text_num = g_strdup_printf("<span font_desc='128'>%d</span>", num++);
    if (num == 10)
        num = 0;
    gtk_label_set_markup(GTK_LABEL(label), text_num);
    return TRUE;
}

/*
 * refresh_sum - Refresh the num in window3
```

```
 */
gboolean refresh_sum(gpointer label) {
    static int sum = 0;
    static int add = 1;

    gchar *text_old = g_strdup_printf("<span font_desc='48'>%d+%d=</span>", sum, add);
    sum += add++;
    gchar *text_new = g_strdup_printf("<span font_desc='32'>%d</span>", sum);
    if (add == 1000) {
        sum = 0;
        add = 1;
    }

    gchar *text_data = g_strdup_printf("\n%s\n\n%s\n", text_old, text_new);
    gtk_label_set_markup(GTK_LABEL(label), text_data);
    return TRUE;
}

int main(int argc, char **argv) {
    int pid1, pid2;
    int wait_tmp;

    switch(pid1 = fork()) {

    case -1:
        err_exit("Fork child 1");

    /* Child pid 1 - Show time */
    case 0:
        gtk_init(&argc, &argv);
        GtkWidget *window1 = gtk_window_new(GTK_WINDOW_TOPLEVEL);
        g_signal_connect(G_OBJECT(window1), "delete_event", G_CALLBACK(gtk_main_quit), NULL);
        gtk_window_set_title(GTK_WINDOW(window1), "Window1: Show Time");

        /* Window1 inside */
        GtkWidget *label1 = gtk_label_new(NULL);
        gtk_container_add(GTK_CONTAINER(window1), label1);
        /*
          * g_timeout_add - Sets a function to be called at regular intervals,
          * with the default priority, G_PRIORITY_DEFAULT.
          *
          * 1000: the time between calls to the function, in milliseconds
          */
        gint s1 = g_timeout_add(RE_FEQ, refresh_time, (void *)label1);

        gtk_widget_set_size_request(window1, WINDOW_WIDTH, WINDOW_HEIGHT);
        gtk_widget_show_all(window1);
        gtk_main();
        printf("Window 1 Closed!\n");
        exit(0);

    default:
        switch(pid2 = fork()) {

        case -1:
            err_exit("Fork child 2");
```

```
    /* Child pid 2 - Show 0 to 9 */
    case 0:
        gtk_init(&argc, &argv);
        GtkWidget *window2 = gtk_window_new(GTK_WINDOW_TOPLEVEL);
        g_signal_connect(G_OBJECT(window2), "delete_event", G_CALLBACK(gtk_main_quit), NULL);
        gtk_window_set_title(GTK_WINDOW(window2), "Window2: Show 0 ~ 9");

        /* Window2 inside */
        GtkWidget *label2 = gtk_label_new(NULL);
        gtk_container_add(GTK_CONTAINER(window2), label2);
        gint s2 = g_timeout_add(RE_FEQ, refresh_num, (void *)label2);

        gtk_widget_set_size_request(window2, WINDOW_WIDTH, WINDOW_HEIGHT);
        gtk_widget_show_all(window2);
        gtk_main();
        printf("Window 2 Closed!\n");
        exit(0);

    /* Parent pid - Add 1 to 1000 */
    default:
        gtk_init(&argc, &argv);
        GtkWidget *window3 = gtk_window_new(GTK_WINDOW_TOPLEVEL);
        g_signal_connect(G_OBJECT(window3), "delete_event", G_CALLBACK(gtk_main_quit), NULL);
        gtk_window_set_title(GTK_WINDOW(window3), "Window3: Show Add 1 to 1000");

        /* Window3 inside */
        GtkWidget *label3 = gtk_label_new(NULL);
        gtk_container_add(GTK_CONTAINER(window3), label3);
        gint s3 = g_timeout_add(RE_FEQ, refresh_sum, (void *)label3);

        gtk_widget_set_size_request(window3, WINDOW_WIDTH, WINDOW_HEIGHT);
        gtk_widget_show_all(window3);
        gtk_main();
        /* Wait for child pid */
        printf("Window 3 Closed!\n");
        waitpid(pid1, &wait_tmp, 0);
        waitpid(pid1, &wait_tmp, 0);
        exit(0);
    }
  }
}
```

设计内容二 增加系统调用

sys.c – 增加的系统调用（仅添加的部分）

```
/*
 * HUST OS Design - Part II
 *
 * Syscall - testcall, teststr, mycopy
 *
 * Added by zxcpyp at 2018-07-22
 *
 * Github: zxc479773533
 */
```

```
SYSCALL_DEFINE0(testcall) {
        printk(KERN_INFO "Hello world!");
        return 0;
}

SYSCALL_DEFINE1(teststr, const char *, str) {
        char buf[256];
        long read_num;

        /* Set memory access range */
        mm_segment_t fs = get_fs();
        set_fs(get_ds());

        /* Copy str form user space to kernel space */
        read_num = strncpy_from_user(buf, str, sizeof(buf));
        if (read_num < 0 || read_num == sizeof(buf)) {
                set_fs(fs);
                return -EFAULT;
        }

        printk(KERN_INFO "System call teststr: %s", buf);
        set_fs(fs);
        return 0;
}

SYSCALL_DEFINE2(mycopy, const char *, s_file, const char *, t_file) {
        struct kstat k_buf;
        char copy_buf[1024];
        char s_filename[256], t_filename[256];
        int read_fd, write_fd;
        long read_num;

        /* Set memory access range */
        mm_segment_t fs = get_fs();
        set_fs(get_ds());

        /* Get source and target file name */
        read_num = strncpy_from_user(s_filename, s_file, sizeof(s_filename));
        if (read_num < 0 || read_num == sizeof(s_filename)) {
                set_fs(fs);
                return -EFAULT;
        }
        read_num = strncpy_from_user(t_filename, t_file, sizeof(t_filename));
        if (read_num < 0 || read_num == sizeof(t_filename)) {
                set_fs(fs);
                return -EFAULT;
        }

        /* Get source file mode */
        if (vfs_stat(s_filename, &k_buf) != 0) {
                set_fs(fs);
                return -EFAULT;
        }

        /* Open files */
        if ((read_fd = ksys_open(s_filename, O_RDONLY, S_IRUSR)) == -1) {
                set_fs(fs);
```

```
                return -EFAULT;
        }
        printk("After open source file");
        if ((write_fd = ksys_open(t_filename, O_WRONLY | O_CREAT | O_TRUNC, k_buf.mode)) ==
-1) {
                set_fs(fs);
                return -EFAULT;
        }

        /* Do copy */
        for (;;) {
                read_num = ksys_read(read_fd, copy_buf, sizeof(copy_buf));
                if (read_num < 0) {
                        set_fs(fs);
                        return -EFAULT;
                } else if (read_num == 0)
                        break;
                ksys_write(write_fd, copy_buf, read_num);
        }

        ksys_close(read_fd);
        ksys_close(write_fd);

        /* Restore previous memory access */
        set_fs(fs);

        return 0;
}
```

testcall.c – 测试系统调用

```
/*
 * HUST OS Design - Part II
 *
 * Syscall - test syscall testcall
 *
 * Created by zxcpyp at 2018-07-22
 *
 * Github: zxc479773533
 */

#include <unistd.h>
#include <stdio.h>
#include <errno.h>

int main(void) {
    printf("/*\n");
    printf(" * HUST OS Design - Part II\n");
    printf(" *\n");
    printf(" * Test01: Test syscall\n");
    printf(" * issue: dmesg to see \"Hello world!\"\n");
    printf(" */\n");
    syscall(333);
    return 0;
}
```

teststr.c – 测试系统调用传参

```
/*
 * HUST OS Design - Part II
 *
 * Syscall - test syscall teststr
 *
 * Created by zxcpyp at 2018-07-22
 *
 * Github: zxc479773533
 */

#include <unistd.h>
#include <stdio.h>
#include <errno.h>

int main(int argc, char **argv) {
    printf("/*\n");
    printf(" * HUST OS Design - Part II\n");
    printf(" *\n");
    printf(" * Test02: Test string pass\n");
    printf(" * issue: dmesg to see your input string\n");
    printf(" */\n\n");
    if (argc == 1) {
        printf("\nUsage: ./teststr <str>\n");
        return 0;
    }
    printf("Arg: %s\n", argv[1]);
    long ret = syscall(334, argv[1]);
    printf("ret: %ld\n", ret);
    printf("errno: %d\n", errno);
    return 0;
}
```

testcp.c – 测试系统调用文件拷贝

```
/*
 * HUST OS Design - Part II
 *
 * Syscall - test syscall mycopy
 *
 * Created by zxcpyp at 2018-07-22
 *
 * Github: zxc479773533
 */

#include <unistd.h>
#include <stdio.h>
#include <errno.h>

int main(int argc, char **argv) {
    printf("/*\n");
    printf(" * HUST OS Design - Part II\n");
    printf(" *\n");
```

```
    printf(" * Test03: Test mycopy\n");
    printf(" */\n\n");
    if (argc != 3) {
        printf("Usage ./testcp <src> <dst>\n");
        return 0;
    }
    printf("Copy: %s -> %s\n", argv[1], argv[2]);
    long ret = syscall(335, argv[1], argv[2]);
    printf("ret: %ld\n", ret);
    printf("errno: %d\n", errno);
    return 0;
}
```

设计内容三 增加设备驱动程序

zxcpyp_dev.c – 设备驱动文件

```
/*
 * HUST OS Design - Part III
 *
 * Device driver - Character device driver
 *
 * Created by zxcpyp at 2018-07-27
 *
 * Github: zxc479773533
 */

#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/cdev.h>
#include <linux/string.h>
#include <linux/slab.h>
#include <linux/fs.h>
#include <linux/uaccess.h>

#define ZXCPYPMEM_SIZE 0x4000    /* 8KB for zxcpyp driver */
#define GLOBAL_MAJOR 500    /* Set device major */
#define MEM_CLEAR 1    /* Use for ioctl to clear memory */

MODULE_LICENSE("GPL");
MODULE_AUTHOR("zxcpyp <zxc479773533@gmail.com>");
MODULE_DESCRIPTION("A simple character driver module");

struct zxcpyp_dev {
    struct cdev cdev;
    unsigned char mem[ZXCPYPMEM_SIZE];
};

static struct zxcpyp_dev *zxcpyp_devp;    /* Global dev pointer */
struct device *device;
struct class *class;

/*
 * zxcpypdriver_open - Open the driver
```

```
 */
static int zxcpypdriver_open(struct inode *inodep, struct file *filep) {
   printk(KERN_INFO "ZXCPYP Driver: open\n");
   filep->private_data = zxcpyp_devp;
   return 0;
}

/*
 * zxcpypdriver_release - Release the driver
 */
static int zxcpypdriver_release(struct inode *inodep, struct file *filep) {
   printk(KERN_INFO "ZXCPYP Driver: release\n");
   return 0;
}

/*
 * zxcpypdriver_read - Read from the driver
 */
static ssize_t zxcpypdriver_read(struct file *filep, char __user *buf, size_t count, loff_t *offset) {
   printk(KERN_INFO "ZXCPYP Driver: start read\n");

   int ret = 0;
   size_t avail = ZXCPYPMEM_SIZE - *offset;
   struct zxcpyp_dev *dev = filep->private_data;

   /* Available memory exists */
   if (count <= avail) {
      if (copy_to_user(buf, dev->mem + *offset, count) != 0)
         return -EFAULT;
      *offset += count;
      ret = count;
   }
   /* Available memory not enough */
   else {
      if (copy_to_user(buf, dev->mem + *offset, avail) != 0)
         return -EFAULT;
      *offset += avail;
      ret = avail;
   }

   printk(KERN_INFO "ZXCPYP Driver: read %u bytes\n", ret);
   return ret;
}

/*
 * zxcpypdriver_write - Write from the driver
 */
static ssize_t zxcpypdriver_write(struct file *filep, const char __user *buf, size_t count, loff_t *offset) {
   printk(KERN_INFO "ZXCPYP Driver: start write\n");

   int ret = 0;
   size_t avail = ZXCPYPMEM_SIZE - *offset;
   struct zxcpyp_dev *dev = filep->private_data;
   memset(dev->mem + *offset, 0, avail);
   printk(KERN_INFO "ZXCPYP Driver: After write\n");

   /* Available memory exists */
```

```
    if (count > avail) {
        if (copy_from_user(dev->mem + *offset, buf, avail) != 0)
            return -EFAULT;
        *offset += avail;
        ret = avail;
    }
    /* Available memory not enough */
    else {
        if (copy_from_user(dev->mem + *offset, buf, count) != 0)
            return -EFAULT;
        *offset += count;
        ret = count;
    }

    printk(KERN_INFO "ZXCPYP Driver: written %u bytes\n", ret);
    return ret;
}

/*
  * zxcpypdriver_llseek - Set the current position of the file for reading and writing
  */
static loff_t zxcpypdriver_llseek(struct file *filep, loff_t offset, int whence) {
    printk(KERN_INFO "ZXCPYP Driver: start llseek\n");

    loff_t ret = 0;
    switch (whence) {
    /* SEEK_SET */
    case 0:
        if (offset < 0) {
            ret = -EINVAL;
            break;
        }
        if (offset > ZXCPYPMEM_SIZE) {
            ret = -EINVAL;
            break;
        }
        ret = offset;
        break;
    /* SEEK_CUR*/
    case 1:
        if ((filep->f_pos + offset) > ZXCPYPMEM_SIZE) {
            ret = -EINVAL;
            break;
        }
        if ((filep->f_pos + offset) < 0) {
            ret = -EINVAL;
            break;
        }
        ret = filep->f_pos + offset;
        break;
    /*
      * SEEK_END: Here we can't use SEEK_END,
      *                 beacuse the memory is solid.
      *
    case 2:
        if (offset < 0) {
            ret = -EINVAL;
```

```
            break;
        }
        ret = ZXCPYPMEM_SIZE + offset;
        break;*/
    /* Else: return error */
    default:
        ret = -EINVAL;
    }

    if (ret < 0)
        return ret;

    printk(KERN_INFO "ZXCPYP Driver: set offset to %u\n", ret);
    filep->f_pos = ret;
    return ret;
}

/*
 * zxcpypdriver_ioctl - Control the zxcpyp driver(memory clear)
 */
static long zxcpypdriver_ioctl(struct file *filep, unsigned int cmd, unsigned long arg) {
    printk(KERN_INFO "ZXCPYP Driver: start memory clear\n");

    struct zxcpyp_dev *dev = filep->private_data;
    switch (cmd) {
    case MEM_CLEAR:
        memset(dev->mem, 0, ZXCPYPMEM_SIZE);
        printk("ZXCPYP Driver: memory is set to zero\n");
        break;
    default:
        return -EINVAL;
    }
    return 0;
}

/*
 * Set operation pointers
 */
static const struct file_operations fops = {
    .owner = THIS_MODULE,
    .open = zxcpypdriver_open,
    .release = zxcpypdriver_release,
    .read = zxcpypdriver_read,
    .write = zxcpypdriver_write,
    .llseek = zxcpypdriver_llseek,
    .unlocked_ioctl = zxcpypdriver_ioctl,
};

/*
 * zxcpypdriver_init - Initial function for zxcpypdriver
 */
static int __init zxcpypdriver_init(void) {
    printk(KERN_INFO "Load module: zxcpypdriver\n");

    int ret;
    dev_t devno = MKDEV(GLOBAL_MAJOR, 0);
    ret = register_chrdev_region(devno, 1, "zxcpypdriver");
```

```
    if (ret < 0) {
        printk(KERN_ALERT "Registering the character device failed with %d\n", ret);
        return ret;
    }

    /* Alloc memory for device */
    zxcpyp_devp = kzalloc(sizeof(struct zxcpyp_dev), GFP_KERNEL);
    if (zxcpyp_devp == NULL) {
        printk(KERN_ALERT "Alloc memory for device failed\n");
        ret = -ENOMEM;
        goto failed;
    }
    memset(zxcpyp_devp->mem, 0, ZXCPYPMEM_SIZE);

    /* Setup device */
    cdev_init(&zxcpyp_devp->cdev, &fops);
    zxcpyp_devp->cdev.owner = THIS_MODULE;
    cdev_add(&zxcpyp_devp->cdev, devno, 1);

    /* Creat device file */
    class = class_create(THIS_MODULE, "zxcpypdriver");
    if (IS_ERR(class)) {
        ret = PTR_ERR(class);
        printk(KERN_ALERT "Creat class for device file failed with %d\n", ret);
        goto failed;
    }
    device = device_create(class, NULL, devno, NULL, "zxcpypdriver");
    if (IS_ERR(device)) {
        class_destroy(class);
        ret = PTR_ERR(device);
        printk(KERN_ALERT "Creat device file failed with %d\n", ret);
        goto failed;
    }

    return 0;

  failed:
    unregister_chrdev_region(devno, 1);
    return ret;
}

/*
  * zxcpypdriver_exit - Exit function for zxcpypdriver
  */
static void __exit zxcpypdriver_exit(void) {
    printk(KERN_INFO "Unload module: zxcpypdriver\n");

    device_destroy(class, MKDEV(GLOBAL_MAJOR, 0));
    class_unregister(class);
    class_destroy(class);

    cdev_del(&zxcpyp_devp->cdev);
    kfree(zxcpyp_devp);
    unregister_chrdev_region(MKDEV(GLOBAL_MAJOR, 0), 1);
}

module_init(zxcpypdriver_init);
```

```
module_exit(zxcpypdriver_exit);
```

Makefile – 编译设备驱动

```
# Kernel lib
KVERS :=$(shell uname -r)

# Path
PWD :=$(shell pwd)

# Build modules
obj-m += zxcpyp_dev.o

build:
	make -C /lib/modules/$(KVERS)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(KVERS)/build M=$(PWD) clean
```

testdev.c – 测试设备驱动

```
/*
 * HUST OS Design - Part III
 *
 * Device driver - Character device driver
 *
 * Created by zxcpyp at 2018-07-28
 *
 * Github: zxc479773533
 */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

/* Cmd for ioctl */
#define MEM_CLEAR 1

/*
 * print_usage - Print usage of testdev
 */
void print_usage(void) {
    printf("Usage: \n");
    printf("\t./testdev read <startpos> <readnum>\n");
    printf("\t./testdev write <startpos> <string>\n");
    printf("\t./testdev ioctl clear\n");
}

int main(int argc, char **argv) {
    printf("/*\n");
```

```
    printf(" * HUST OS Design - Part III\n");
    printf(" *\n");
    printf(" * Test Device: zxcpypdriver\n");
    printf(" */\n\n");
    int fd, start, num;
    char buf[1024];
    fd = open("/dev/zxcpypdriver", O_RDWR);
    if (fd < 0) {
        printf("Open error!\n");
        return 0;
    }
    if (argc == 4 && strncmp(argv[1], "read", 4) == 0) {
        start = atoi(argv[2]);
        num = atoi(argv[3]);
        lseek(fd, start, SEEK_SET);
        read(fd, buf, num);
        printf("Read: %s\n", buf);
    }
    else if (argc == 4 && strncmp(argv[1], "write", 5) == 0) {
        start = atoi(argv[2]);
        lseek(fd, start, SEEK_CUR);
        write(fd, argv[3], strlen(argv[3]));
        printf("Write succeed!\n");
    }
    else if (argc == 3 && strncmp(argv[1], "ioctl", 5) == 0) {
        if (strncmp(argv[2], "clear", 5) == 0) {
            ioctl(fd, MEM_CLEAR, NULL);
            printf("Clear success!\n");
        }
        else
            print_usage();
    }
    else
        print_usage();

    close(fd);
    return 0;
}
```

设计内容四 系统监控器

sysmonitor.h – 系统监控器头文件

```
/*
 * HUST OS Design - Part IV
 *
 * System Monitor - Header
 *
 * Created by zxcpyp at 2018-08-21
 *
 * Github: zxc479773533
 */

#include "../lib/zxcpyp_sys.h"
#include <dirent.h>
```

```
#include <gtk/gtk.h>

/* Set author face picture */
#define AUTHOR "../PartIV-System_monitor/data/author.jpg"

/* Global variables */
GtkWidget *popup_window;
GtkWidget *popup_label;
char *now_pid = NULL;            /* Hold the pid clicked by mouse */
gint process_num = 0;            /* Hold the Process num */
GtkWidget *clist;                /* Hold the Process info */
GtkWidget *clist2;               /* Hold the Modules info */
GtkWidget *entry;                /* Hold the input */
float cpu_ratio = 0;             /* Hold the CPU use ratio */
float cpu_ratio_data[120];    /* Hold the CPU use ratio histories */
char cpu_freq[1024];             /* Hold the CPU frequence */
float mem_total = 0;             /* Hold the Memory total size */
float mem_free = 0;              /* Hold the Memory free size */
float mem_ratio = 0;             /* Hold the Memory use ratio */
float mem_ratio_data[120];    /* Hold the Memory use ratio histories */
float swap_total = 0;            /* Hold the Swap total size */
float swap_free = 0;             /* Hold the Swap free size */
float swap_ratio = 0;            /* Hold the Swap use ratio */
float swap_ratio_data[120]; /* Hold the Swap use ratio histories */
GtkWidget *cpu_curve;            /* Hold the CPU use curve */
GtkWidget *mem_curve;            /* Hold the Memory use curve */
GtkWidget *swap_curve;           /* Hold the Swap use curve */
int cpu_curve_start = 20;     /* Hold the CPU use curve start position */
int mem_curve_start = 20;     /* Hold the CPU use curve start position */
int swap_curve_start = 20;    /* Hold the CPU use curve start position */
float receive_speed = 0;      /* Hold the Network receive speed */
float send_speed = 0;            /* Hold the Network send speed */
float read_speed = 0;            /* Hold the Disk read speed */
float write_speed = 0;           /* Hold the Disk write speed */

/* Callback functions */
void select_row_callback(GtkWidget *clist, gint row, gint column, GdkEventButton *event, gpointer data);
void search_proc(GtkButton *button,gpointer data);
void kill_proc(void);
void refresh_proc(void);
gboolean cpu_curve_callback(GtkWidget *widget, GdkEventExpose *event, gpointer data);
gboolean mem_curve_callback(GtkWidget *widget, GdkEventExpose *event, gpointer data);
gboolean swap_curve_callback(GtkWidget *widget, GdkEventExpose *event, gpointer data);
void refresh_modules(void);

/* Loop functions */
gboolean draw_cpu_curve(gpointer widget);
gboolean draw_mem_curve(gpointer widget);
gboolean draw_swap_curve(gpointer widget);
gboolean get_cpu_ratio(gpointer label);
gboolean get_cpu_mhz(gpointer label);
gboolean get_memory_ratio(gpointer label);
gboolean get_memory_fraction(gpointer label);
gboolean get_swap_ratio(gpointer label);
gboolean get_swap_fraction(gpointer label);
gboolean get_sys_info(gpointer label);
gboolean get_network_info(gpointer label);
```

```
gboolean get_disk_info(gpointer label);

/* Assist functions */
char *utf8_fix(char *c);
void scroll_to_line(gpointer scrolled_window, gint line_num, gint to_line_index);
void set_label_fontsize(GtkWidget *label, char *fontsize);
```

sysmonitor.h – 系统监控器实现

```
/*
 * HUST OS Design - Part IV
 *
 * System Monitor - Implementation
 *
 * Created by zxcpyp at 2018-08-21
 *
 * Github: zxc479773533
 */

#include "sysmonitor.h"

/*
 * read_stat - Read data from /proc/pid/stat
 */
void read_stat(char (*info)[1024], char *stat_file) {
    /*
     * stat file format:
     *
     * pid (name) status ppid .....(13 data) priority (4 data) memory
     */
    int pos;

    /* Get pid */
    for (pos = 0; pos < 1024; pos++) {
        if (stat_file[pos] == ' ')
            break;
    }
    stat_file[pos] = '\0';
    strcpy(info[0], stat_file);
    stat_file += pos;
    stat_file += 2;

    /* Get name */
    for (pos = 0; pos < 1024; pos++) {
        if (stat_file[pos] == ')')
            break;
    }
    stat_file[pos] = '\0';
    strcpy(info[1], stat_file);
    stat_file += pos;
    stat_file += 2;

    /* Get status */
    for (pos = 0; pos < 1024; pos++) {
        if (stat_file[pos] == ' ')
```

```
            break;
        }
        stat_file[pos] = '\0';
        strcpy(info[3], stat_file);
        stat_file += pos;
        stat_file += 1;

        /* Get ppid */
        for (pos = 0; pos < 1024; pos++) {
            if (stat_file[pos] == ' ')
                break;
        }
        stat_file[pos] = '\0';
        strcpy(info[2], stat_file);
        stat_file += pos;
        stat_file += 1;

        /* Get priority */
        int i;
        for (i = 0, pos = 0; pos < 1024; pos++) {
            if (stat_file[pos] == ' ')
                i++;
            if (i == 13)
                break;
        }
        stat_file[pos] = '\0';
        stat_file += pos;
        stat_file += 1;
        for (pos = 0; pos < 1024; pos++) {
            if (stat_file[pos] == ' ')
                break;
        }
        stat_file[pos] = '\0';
        strcpy(info[4], stat_file);
        stat_file += pos;
        stat_file += 1;

        /* Get memory use */
        for (i = 0, pos = 0; pos < 1024; pos++) {
            if (stat_file[pos] == ' ')
                i++;
            if (i == 4)
                break;
        }
        stat_file[pos] = '\0';
        stat_file += pos;
        stat_file += 1;
        for (pos = 0; pos < 1024; pos++) {
            if (stat_file[pos] == ' ')
                break;
        }
        stat_file[pos] = '\0';
        char buf[1024];
        sprintf(buf, "%d KB\0", abs(atoi(stat_file)) / 1024);
        strcpy(info[5], buf);
}
```

```
/*
  * get_process_info - Get process info in /proc and add into clist
  */
void get_process_info(void) {
    DIR *dir;
    struct dirent *dir_info;
    int fd;
    char pid_file[1024];
    char stat_file[1024];
    char *one_file = NULL;
    char info[6][1024];
    gchar *txt[6];

    /* Set clist column title */
    gtk_clist_set_column_title(GTK_CLIST(clist), 0, "PID");
    gtk_clist_set_column_title(GTK_CLIST(clist), 1, "Name");
    gtk_clist_set_column_title(GTK_CLIST(clist), 2, "PPID");
    gtk_clist_set_column_title(GTK_CLIST(clist), 3, "State");
    gtk_clist_set_column_title(GTK_CLIST(clist), 4, "Priority");
    gtk_clist_set_column_title(GTK_CLIST(clist), 5, "Memory use");

    /* Set clist column width */
    gtk_clist_set_column_width(GTK_CLIST(clist), 0, 75);
    gtk_clist_set_column_width(GTK_CLIST(clist), 1, 125);
    gtk_clist_set_column_width(GTK_CLIST(clist), 2, 75);
    gtk_clist_set_column_width(GTK_CLIST(clist), 3, 75);
    gtk_clist_set_column_width(GTK_CLIST(clist), 4, 75);
    gtk_clist_set_column_width(GTK_CLIST(clist), 5, 125);
    gtk_clist_column_titles_show(GTK_CLIST(clist));

    /* Read proc info */
    dir = opendir("/proc");
    process_num = 0;
    while (dir_info = readdir(dir)) {
        /*
          * If start with number, then read it
          */
        if ((dir_info->d_name)[0] >= '0' && ((dir_info->d_name)[0]) <= '9') {
            sprintf(pid_file, "/proc/%s/stat", dir_info->d_name);
            fd = open(pid_file, O_RDONLY);
            read(fd, stat_file, 1024);
            close(fd);
            one_file = stat_file;
            read_stat(info, one_file);
            for (int i = 0; i < 6; i++)
                txt[i] = utf8_fix(info[i]);
            gtk_clist_append(GTK_CLIST(clist), txt);
            process_num++;
        }
    }
    closedir(dir);
}

void get_modules_info(void) {
    FILE *fp;
    char modules_info[1024];
    char *line = NULL;
```

```
    char info[3][1024];
    gchar *txt[3];
    int pos = 0;

    /* Set clist column title */
    gtk_clist_set_column_title(GTK_CLIST(clist2), 0, "Module name");
    gtk_clist_set_column_title(GTK_CLIST(clist2), 1, "Memory use");
    gtk_clist_set_column_title(GTK_CLIST(clist2), 2, "Used Times");

    /* Set clist column width */
    gtk_clist_set_column_width(GTK_CLIST(clist2), 0, 250);
    gtk_clist_set_column_width(GTK_CLIST(clist2), 1, 150);
    gtk_clist_set_column_width(GTK_CLIST(clist2), 2, 150);
    gtk_clist_column_titles_show(GTK_CLIST(clist2));

    fp = fopen("/proc/modules", "r");

    while ((line = fgets(modules_info, sizeof(modules_info), fp)) != NULL) {
        /* Read modules name */
        for (pos = 0; pos < 1024; pos++) {
            if (line[pos] == ' ')
                break;
        }
        line[pos] = '\0';
        strcpy(info[0], line);
        pos++;
        line += pos;

        /* Read modules memory use */
        for (pos = 0; pos < 1024; pos++) {
            if (line[pos] == ' ')
                break;
        }
        line[pos] = '\0';
        strcpy(info[1], line);
        pos++;
        line += pos;

        /* Read modules name */
        for (pos = 0; pos < 1024; pos++) {
            if (line[pos] == ' ')
                break;
        }
        line[pos] = '\0';
        strcpy(info[2], line);

        for (int i = 0; i < 3; i++)
            txt[i] = utf8_fix(info[i]);

        gtk_clist_append(GTK_CLIST(clist2), txt);
    }
    fclose(fp);
}

/*
 * get_cpu_info - get cpu information from /proc/cpuinfo
 */
```

```
void get_cpu_info(char *cpu_name,
                  char *cpu_addr_digit,
                  char *cpu_cache_size,
                  char *cpu_core) {
    /*
     * cpuinfo file format:
     *
     * model name: line 5
     * cache size: line 9
     * cpu cores: line 13
     * address sizes: line 25
     */
    int fd;
    char info_buf[2048];    /* Here 1024 is too small */
    char info_str[1024];
    char *pos = NULL;
    int i;
    fd = open("/proc/cpuinfo", O_RDONLY);
    read(fd, info_buf, sizeof(info_buf));
    close(fd);

    /* Read CPU Name */
    i = 0;
    pos = strstr(info_buf, "model name");
    while (*pos != ':')
        pos++;
    pos += 2;

    while (*pos != '\n') {
        info_str[i] = *pos;
        i++;
        pos++;
    }
    info_str[i] = '\0';
    strcpy(cpu_name, info_str);

    /* Read Cache size */
    i = 0;
    pos = strstr(info_buf, "cache size");
    while (*pos != ':')
        pos++;
    pos += 2;
    while (*pos != '\n') {
        info_str[i] = *pos;
        i++;
        pos++;
    }
    info_str[i] = '\0';
    strcpy(cpu_cache_size, info_str);

    /* Read CPU Cores */
    i = 0;
    pos = strstr(info_buf, "cpu cores");
    while (*pos != ':')
        pos++;
    pos += 2;
    while (*pos != '\n') {
```

```
        info_str[i] = *pos;
        i++;
        pos++;
    }
    info_str[i] = '\0';
    strcpy(cpu_core, info_str);

    /* Read Addressing digital */
    i = 0;
    pos = strstr(info_buf, "address sizes");
    while (*pos != ':')
        pos++;
    pos += 2;
    while (*pos != '\n') {
        info_str[i] = *pos;
        i++;
        pos++;
    }
    info_str[i] = '\0';
    strcpy(cpu_addr_digit, info_str);
}


int main(int argc, char **argv) {
    gtk_init(&argc, &argv);
    /* GTK widgets */
    GtkWidget *top_window;
    GtkWidget *notebook;
    GtkWidget *hbox;
    GtkWidget *vbox;
    GtkWidget *label;
    GtkWidget *label1;
    GtkWidget *label2;
    GtkWidget *label3;
    GtkWidget *scrolled_window;
    GtkWidget *frame;
    GtkWidget *cpu_use;
    GtkWidget *mem_use;
    GtkWidget *swap_use;
    GtkWidget *button1;
    GtkWidget *button2;
    GtkWidget *button3;
    GtkWidget *fixed;
    GtkWidget *image;

    /* Save page title */
    char title_buf[1024];

    /* Buffer to use */
    char buffer1[1024];
    char cpu_name[1024];
    char cpu_addr_digit[1024];
    char cpu_cache_size[1024];
    char cpu_core[1024];

    /* Create top window */
    top_window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
```

```
    gtk_window_set_title(GTK_WINDOW(top_window), "System monitor");
    gtk_widget_set_size_request(top_window, 600, 700);
    g_signal_connect(G_OBJECT(top_window), "delete_event", G_CALLBACK(gtk_main_quit), NULL);
    gtk_container_set_border_width(GTK_CONTAINER(top_window), 10);

    /* Create notebook */
    notebook = gtk_notebook_new();
    gtk_container_add(GTK_CONTAINER(top_window), notebook);
    gtk_notebook_set_tab_pos(GTK_NOTEBOOK(notebook), GTK_POS_TOP);


    /*
      * Page 1: Process manage
      */
    sprintf(title_buf, "Process");
    vbox = gtk_vbox_new(FALSE, 0);

    /* Create scrolled window for process info */
    scrolled_window = gtk_scrolled_window_new(NULL, NULL);
    gtk_widget_set_size_request(scrolled_window, 550, 500);
    gtk_scrolled_window_set_policy(GTK_SCROLLED_WINDOW(scrolled_window),
GTK_POLICY_AUTOMATIC, GTK_POLICY_AUTOMATIC);

    /* Create list with 6 columus */
    clist = gtk_clist_new(6);
    get_process_info();
    gtk_signal_connect(GTK_OBJECT(clist), "select_row", GTK_SIGNAL_FUNC(select_row_callback),
NULL);
    gtk_scrolled_window_add_with_viewport(GTK_SCROLLED_WINDOW(scrolled_window), clist);
    gtk_box_pack_start(GTK_BOX(vbox), scrolled_window, TRUE, TRUE, 5);

    /* Create buttons */
    hbox = gtk_hbox_new(FALSE, 10);
    entry = gtk_entry_new();
    gtk_entry_set_max_length(GTK_ENTRY(entry), 0);
    button1 = gtk_button_new_with_label("Search by PID");
    button2 = gtk_button_new_with_label("Kill");
    button3 = gtk_button_new_with_label("Refresh");
    g_signal_connect(G_OBJECT(button1), "clicked", G_CALLBACK(search_proc), scrolled_window);
    g_signal_connect(G_OBJECT(button2), "clicked", G_CALLBACK(kill_proc), NULL);
    g_signal_connect(G_OBJECT(button3), "clicked", G_CALLBACK(refresh_proc), NULL);
    gtk_widget_set_size_request(entry, 200, 30);
    gtk_widget_set_size_request(button1, 120, 30);
    gtk_widget_set_size_request(button2, 80, 30);
    gtk_widget_set_size_request(button3, 80, 30);
    gtk_box_pack_start(GTK_BOX(hbox), entry, FALSE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox), button1, FALSE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox), button2, FALSE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox), button3, FALSE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, FALSE, 5);
    label = gtk_label_new(title_buf);
    gtk_notebook_append_page(GTK_NOTEBOOK(notebook), vbox, label);


    /*
      * Page 2: CPU info
      */
```

```
    sprintf(title_buf, "CPU");
    vbox = gtk_vbox_new(FALSE, 0);

    /* Creat frame to show curve of CPU use */
    hbox = gtk_hbox_new(FALSE, 0);
    gtk_box_pack_start(GTK_BOX(vbox), hbox, TRUE, FALSE, 5);
    cpu_use = gtk_frame_new("CPU Use");
    gtk_container_set_border_width(GTK_CONTAINER(cpu_use), 5);
    gtk_widget_set_size_request(cpu_use, 520, 300);
    gtk_box_pack_start(GTK_BOX(hbox), cpu_use, TRUE, FALSE, 5);

    hbox = gtk_hbox_new(FALSE, 0);
    label1 = gtk_label_new("");
    label2 = gtk_label_new("");
    gtk_box_pack_start(GTK_BOX(hbox), label1, TRUE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox), label2, TRUE, FALSE, 5);
    g_timeout_add(1000, (GtkFunction)get_cpu_ratio, (gpointer)label1);
    g_timeout_add(1000, (GtkFunction)get_cpu_mhz, (gpointer)label2);
    gtk_widget_set_size_request(hbox, 550, 30);
    gtk_box_pack_start(GTK_BOX(vbox), hbox, TRUE, FALSE, 5);

    /* Draw CPU use curve */
    cpu_curve = gtk_drawing_area_new();
    gtk_widget_set_size_request(cpu_curve, 0, 0);
    g_signal_connect(G_OBJECT(cpu_curve), "expose_event", G_CALLBACK(cpu_curve_callback),
NULL);
    gtk_container_add(GTK_CONTAINER(cpu_use), cpu_curve);

    /* Creat frame to show CPU info */
    frame = gtk_frame_new("CPU Information");
    gtk_widget_set_size_request(frame, 500, 200);
    gtk_box_pack_start(GTK_BOX(vbox), frame, TRUE, TRUE, 5);
    get_cpu_info(cpu_name, cpu_addr_digit, cpu_cache_size, cpu_core);
    sprintf(buffer1, "CPU Type and Frequency :\n%s\n\nAddressing digit : %s\n\nCache size : %s\n\nCPU
Core : %s\n",
                                cpu_name, cpu_addr_digit, cpu_cache_size, cpu_core);
    label = gtk_label_new(buffer1);
    set_label_fontsize(label, "12");
    gtk_container_add(GTK_CONTAINER(frame), label);

    label = gtk_label_new(title_buf);
    gtk_notebook_append_page(GTK_NOTEBOOK(notebook), vbox, label);


    /*
      * Page 3: Memory info
      */
    sprintf(title_buf, "Memory");
    vbox = gtk_vbox_new(FALSE, 0);

    /* Create frame to show curve of memory use */
    hbox = gtk_hbox_new(FALSE, 0);
    gtk_box_pack_start(GTK_BOX(vbox), hbox, TRUE, FALSE, 5);
    mem_use = gtk_frame_new("Memory Use");
    gtk_container_set_border_width(GTK_CONTAINER(mem_use), 5);
    gtk_widget_set_size_request(mem_use, 520, 250);
    gtk_box_pack_start(GTK_BOX(hbox), mem_use, TRUE, FALSE, 5);
```

```
    hbox = gtk_hbox_new(FALSE, 0);
    label1 = gtk_label_new("");
    label2 = gtk_label_new("");
    gtk_box_pack_start(GTK_BOX(hbox), label1, TRUE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox), label2, TRUE, FALSE, 5);
    g_timeout_add(1000, (GtkFunction)get_memory_ratio, (gpointer)label1);
    g_timeout_add(1000, (GtkFunction)get_memory_fraction, (gpointer)label2);
    gtk_widget_set_size_request(hbox, 550, 20);
    gtk_box_pack_start(GTK_BOX(vbox), hbox, TRUE, FALSE, 5);

    /* Draw Memory use curve */
    mem_curve = gtk_drawing_area_new();
    gtk_widget_set_size_request(mem_curve, 0, 0);
    g_signal_connect(G_OBJECT(mem_curve), "expose_event", G_CALLBACK(mem_curve_callback),
NULL);
    gtk_container_add(GTK_CONTAINER(mem_use), mem_curve);

    /* Create frame to show curve of swap use */
    hbox = gtk_hbox_new(FALSE, 0);
    gtk_box_pack_start(GTK_BOX(vbox), hbox, TRUE, FALSE, 5);
    swap_use = gtk_frame_new("Swap Use");
    gtk_container_set_border_width(GTK_CONTAINER(swap_use), 5);
    gtk_widget_set_size_request(swap_use, 520, 250);
    gtk_box_pack_start(GTK_BOX(hbox), swap_use, TRUE, FALSE, 5);

    hbox = gtk_hbox_new(FALSE, 0);
    label1 = gtk_label_new("");
    label2 = gtk_label_new("");
    gtk_box_pack_start(GTK_BOX(hbox), label1, TRUE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox), label2, TRUE, FALSE, 5);
    g_timeout_add(1000, (GtkFunction)get_swap_ratio, (gpointer)label1);
    g_timeout_add(1000, (GtkFunction)get_swap_fraction, (gpointer)label2);
    gtk_widget_set_size_request(hbox, 550, 20);
    gtk_box_pack_start(GTK_BOX(vbox), hbox, TRUE, FALSE, 5);

    /* Draw Swap use curve */
    swap_curve = gtk_drawing_area_new();
    gtk_widget_set_size_request(swap_curve, 0, 0);
    g_signal_connect(G_OBJECT(swap_curve), "expose_event", G_CALLBACK(swap_curve_callback),
NULL);
    gtk_container_add(GTK_CONTAINER(swap_use), swap_curve);

    label = gtk_label_new(title_buf);
    gtk_notebook_append_page(GTK_NOTEBOOK(notebook), vbox, label);


    /*
      * Page 4: Modules
      */
    sprintf(title_buf, "Modules");
    vbox = gtk_vbox_new(FALSE, 0);

    /* Create scrolled window for modules info */
    scrolled_window = gtk_scrolled_window_new(NULL, NULL);
    gtk_widget_set_size_request(scrolled_window, 550, 500);
```

```
    gtk_scrolled_window_set_policy(GTK_SCROLLED_WINDOW(scrolled_window),
GTK_POLICY_AUTOMATIC, GTK_POLICY_AUTOMATIC);

    /* Create list with 3 columus */
    clist2 = gtk_clist_new(3);
    get_modules_info();
    gtk_scrolled_window_add_with_viewport(GTK_SCROLLED_WINDOW(scrolled_window), clist2);
    gtk_box_pack_start(GTK_BOX(vbox), scrolled_window, TRUE, TRUE, 5);

    /* Create buttons */
    fixed = gtk_fixed_new();
    button1 = gtk_button_new_with_label("Refresh");
    g_signal_connect(G_OBJECT(button1), "clicked", G_CALLBACK(refresh_modules), NULL);
    gtk_widget_set_size_request(button1, 80, 30);
    gtk_fixed_put(GTK_FIXED(fixed), button1, 450, 0);
    gtk_box_pack_start(GTK_BOX(vbox), fixed, FALSE, FALSE, 5);

    label = gtk_label_new(title_buf);
    gtk_notebook_append_page(GTK_NOTEBOOK(notebook), vbox, label);


    /*
     * Page 5: System info
     */
    sprintf(title_buf, "System");
    vbox = gtk_vbox_new(FALSE, 10);

    frame = gtk_frame_new("System Information");
    label1 = gtk_label_new("");
    gtk_container_add(GTK_CONTAINER(frame), label1);
    g_timeout_add(1000, (GtkFunction)get_sys_info, (gpointer)label1);
    gtk_widget_set_size_request(frame, 550, 300);
    gtk_box_pack_start(GTK_BOX(vbox), frame, TRUE, FALSE, 5);

    frame = gtk_frame_new("Network");
    label2 = gtk_label_new("");
    gtk_container_add(GTK_CONTAINER(frame), label2);
    g_timeout_add(2000, (GtkFunction)get_network_info, (gpointer)label2);
    gtk_widget_set_size_request(frame, 550, 180);
    gtk_box_pack_start(GTK_BOX(vbox), frame, TRUE, FALSE, 0);

    frame = gtk_frame_new("Disk");
    label3 = gtk_label_new("");
    gtk_container_add(GTK_CONTAINER(frame), label3);
    g_timeout_add(2000, (GtkFunction)get_disk_info, (gpointer)label3);
    gtk_widget_set_size_request(frame, 550, 180);
    gtk_box_pack_start(GTK_BOX(vbox), frame, TRUE, FALSE, 0);

    label = gtk_label_new(title_buf);
    gtk_notebook_append_page(GTK_NOTEBOOK(notebook), vbox, label);


    /*
     * Page 5: Author info
     */
    sprintf(title_buf, "About");
    vbox = gtk_vbox_new(FALSE, 0);
```

```
    frame = gtk_frame_new("About Author");
    hbox = gtk_hbox_new(FALSE, 0);
    sprintf(buffer1, "\n\n\n\n%15s: %-25s\n\n%15s: %-25s\n\n%15s: %-25s\n\n\n\n", "Auther", "zxcpyp",
"Github", "zxc479773533", "e-mail", "zxc479773533@gmail.com");
    label = gtk_label_new(buffer1);
    set_label_fontsize(label, "14");
    gtk_box_pack_start(GTK_BOX(hbox), label, TRUE, FALSE, 5);
    image = gtk_image_new_from_file(AUTHOR);
    gtk_box_pack_start(GTK_BOX(hbox), image, TRUE, FALSE, 5);
    gtk_container_add(GTK_CONTAINER(frame), hbox);
    gtk_box_pack_start(GTK_BOX(vbox), frame, TRUE, FALSE, 5);

    frame = gtk_frame_new("About Version");
    sprintf(buffer1, "\n\nVersion v1.2.0\n\n\
Copyright (C) 2018 Pan Yue\n\
School of Computer Science and Technology\n\
Huazhong University of Science and Technology\n\n\n");
    label = gtk_label_new(buffer1);
    set_label_fontsize(label, "14");
    gtk_container_add(GTK_CONTAINER(frame), label);
    gtk_box_pack_start(GTK_BOX(vbox), frame, TRUE, FALSE, 5);

    label = gtk_label_new("For HUST Operating System Curriculum Design 2018 Summer");
    gtk_box_pack_start(GTK_BOX(vbox), label, TRUE, FALSE, 5);

    label = gtk_label_new(title_buf);
    gtk_notebook_append_page(GTK_NOTEBOOK(notebook), vbox, label);

    /* GTK show */
    gtk_widget_show_all(top_window);
    gtk_main();

    return 0;
}


/********************CALLBACKS********************/


/*
  * select_row_callback - Callback for select_row
  */
void select_row_callback(GtkWidget *clist, gint row, gint column, GdkEventButton *event, gpointer data) {
                gtk_clist_get_text(GTK_CLIST(clist), row, 0, &now_pid);
    gtk_entry_set_text(GTK_ENTRY(entry), (gchar *)now_pid);
    return;
}

/*
  * select_row_callback - Search process in GTK clist
  */
void search_proc(GtkButton *button, gpointer data) {
    const gchar *entry_txt;
    gchar *txt;
    gint ret, i = 0;
    entry_txt = gtk_entry_get_text(GTK_ENTRY(entry));
```

```
    while ((ret = gtk_clist_get_text(GTK_CLIST(clist), i, 0, &txt)) != 0) {
        if (!strcmp(entry_txt, txt))
            break;
        i++;
    }
    gtk_clist_select_row(GTK_CLIST(clist), i, 0);
    scroll_to_line(data, process_num, i);
    return;
}

/*
  * select_row_callback - Kill process clicked
  */
void kill_proc(void) {
    int ret;
    if (now_pid != NULL) {
        ret = kill(atoi(now_pid), SIGKILL);
        if (ret == -EPERM) {
            popup_window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
            popup_label = gtk_label_new("You need root privilege\n      to kill this process!");
            set_label_fontsize(popup_label, "14");
            gtk_widget_set_size_request(popup_window, 300, 180);
            gtk_container_add(GTK_CONTAINER(popup_window), popup_label);
            gtk_window_set_title(GTK_WINDOW(popup_window), "ERROR!");
            gtk_widget_show_all(popup_window);
        }
    }
    return;
}

/*
  * refresh_proc - Refresh the process info in clist
  */
void refresh_proc(void) {
    gtk_clist_freeze(GTK_CLIST(clist));
    gtk_clist_clear(GTK_CLIST(clist));
    get_process_info();
    gtk_clist_thaw(GTK_CLIST(clist));
    gtk_clist_select_row(GTK_CLIST(clist), 0, 0);
    return;
}

/*
  * cpu_curve_callback - Refresh the CPU use curve once every second
  */
gboolean cpu_curve_callback(GtkWidget *widget, GdkEventExpose *event, gpointer data) {
    static int flag = 0;
    draw_cpu_curve((gpointer)widget);
    if (flag == 0) {
        g_timeout_add(1000, (GtkFunction)draw_cpu_curve, (gpointer)widget);
        flag = 1;
    }
    return TRUE;
}

/*
  * cpu_curve_callback - Refresh the Memory use curve once every second
```

```
 */
gboolean mem_curve_callback(GtkWidget *widget, GdkEventExpose *event, gpointer data) {
    static int flag = 0;
    draw_mem_curve((gpointer)widget);
    if (flag == 0) {
        g_timeout_add(1000, (GtkFunction)draw_mem_curve, (gpointer)widget);
        flag = 1;
    }
    return TRUE;
}

/*
 * swap_curve_callback - Refresh the Swap use curve once every second
 */
gboolean swap_curve_callback(GtkWidget *widget, GdkEventExpose *event, gpointer data) {
    static int flag = 0;
    draw_swap_curve((gpointer)widget);
    if (flag == 0) {
        g_timeout_add(1000, (GtkFunction)draw_swap_curve, (gpointer)widget);
        flag = 1;
    }
    return TRUE;
}

/*
 * refresh_modules - Refresh the modules info in clist
 */
void refresh_modules(void) {
    gtk_clist_freeze(GTK_CLIST(clist2));
    gtk_clist_clear(GTK_CLIST(clist2));
    get_modules_info();
    gtk_clist_thaw(GTK_CLIST(clist2));
    gtk_clist_select_row(GTK_CLIST(clist2), 0, 0);
    return;
}

/********************LOOPS********************/

/*
 * draw_cpu_curve - Draw CPU curve
 */
gboolean draw_cpu_curve(gpointer widget) {
    GtkWidget *cpu_curve = (GtkWidget *)widget;
    GdkColor color;
    GdkGC *gc = cpu_curve->style->fg_gc[GTK_WIDGET_STATE(widget)];
    static int flag = 0;
    static int now_pos = 0;
    int draw_pos = 0;

    /* Darw background */
    color.red = 0;
    color.green = 0;
    color.blue = 0;
    gdk_gc_set_rgb_fg_color(gc, &color);
    gdk_draw_rectangle(cpu_curve->window, gc, TRUE, 15, 30, 480, 200);

    /* Draw background lines */
```

```
    color.red = 0;
    color.green = 20000;
    color.blue = 0;
    gdk_gc_set_rgb_fg_color(gc, &color);
    for (int i = 30; i <= 220; i += 20)
        gdk_draw_line(cpu_curve->window, gc, 15, i, 495, i);
    for (int i = 15; i <= 480; i += 20)
        gdk_draw_line(cpu_curve->window, gc, i + cpu_curve_start, 30, i + cpu_curve_start, 230);

    /* Settle cpu curve start position to make it live */
    cpu_curve_start -= 4;
    if (cpu_curve_start == 0)
        cpu_curve_start = 20;

    /* Initial data */
    if (flag == 0) {
        for (int i = 0; i < 120; i++) {
            cpu_ratio_data[i] = 0;
            flag = 1;
        }
    }

    /* Add data */
    cpu_ratio_data[now_pos] = cpu_ratio / 100;
    now_pos++;
    if (now_pos == 120)
        now_pos = 0;

    /* Draw lines */
    color.red = 0;
                color.green = 65535;
                color.blue = 0;
                gdk_gc_set_rgb_fg_color(gc, &color);
    draw_pos = now_pos;
    for (int i = 0; i < 119; i++) {
        gdk_draw_line(cpu_curve->window, gc,
                              15 + i * 4, 230 - 200 * cpu_ratio_data[draw_pos % 120],
                              15 + (i + 1) * 4, 230 - 200 * cpu_ratio_data[(draw_pos + 1) % 120]);
        draw_pos++;
        if (draw_pos == 120)
            draw_pos = 0;
    }

    /* Reset the color */
    color.red = 25000;
                color.green = 25000;
                color.blue = 25000;
                gdk_gc_set_rgb_fg_color(gc, &color);

    /* To loop this function, it must return TRUE */
    return TRUE;
}

/*
  * draw_mem_curve - Draw Memory use curve
  */
gboolean draw_mem_curve(gpointer widget) {
```

```
GtkWidget *mem_curve = (GtkWidget *)widget;
GdkColor color;
GdkGC *gc = mem_curve->style->fg_gc[GTK_WIDGET_STATE(widget)];
static int flag = 0;
static int now_pos = 0;
int draw_pos = 0;

/* Darw background */
color.red = 0;
color.green = 0;
color.blue = 0;
gdk_gc_set_rgb_fg_color(gc, &color);
gdk_draw_rectangle(mem_curve->window, gc, TRUE, 15, 10, 480, 200);

/* Draw background lines */
color.red = 0;
color.green = 20000;
color.blue = 0;
gdk_gc_set_rgb_fg_color(gc, &color);
for (int i = 10; i <= 230; i += 20)
   gdk_draw_line(mem_curve->window, gc, 15, i, 495, i);
for (int i = 15; i <= 480; i += 20)
   gdk_draw_line(mem_curve->window, gc, i + mem_curve_start, 10, i + mem_curve_start, 210);

/* Settle memory curve start position to make it live */
mem_curve_start -= 4;
if (mem_curve_start == 0)
   mem_curve_start = 20;

/* Initial data */
if (flag == 0) {
   for (int i = 0; i < 120; i++) {
      mem_ratio_data[i] = 0;
      flag = 1;
   }
}

/* Add data */
mem_ratio_data[now_pos] = mem_ratio / 100;
now_pos++;
if (now_pos == 120)
   now_pos = 0;

/* Draw lines */
color.red = 0;
            color.green = 65535;
            color.blue = 0;
            gdk_gc_set_rgb_fg_color(gc, &color);
draw_pos = now_pos;
for (int i = 0; i < 119; i++) {
   gdk_draw_line(mem_curve->window, gc,
                        15 + i * 4, 210 - 200 * mem_ratio_data[draw_pos % 120],
                        15 + (i + 1) * 4, 210 - 200 * mem_ratio_data[(draw_pos + 1) % 120]);
   draw_pos++;
   if (draw_pos == 120)
      draw_pos = 0;
}
```

```
    /* Reset the color */
    color.red = 25000;
                color.green = 25000;
                color.blue = 25000;
                gdk_gc_set_rgb_fg_color(gc, &color);

    /* To loop this function, it must return TRUE */
    return TRUE;
}

/*
 * draw_swap_curve - Draw Swap use curve
 */
gboolean draw_swap_curve(gpointer widget) {
    GtkWidget *swap_curve = (GtkWidget *)widget;
    GdkColor color;
    GdkGC *gc = swap_curve->style->fg_gc[GTK_WIDGET_STATE(widget)];
    static int flag = 0;
    static int now_pos = 0;
    int draw_pos = 0;

    /* Darw background */
    color.red = 0;
    color.green = 0;
    color.blue = 0;
    gdk_gc_set_rgb_fg_color(gc, &color);
    gdk_draw_rectangle(swap_curve->window, gc, TRUE, 15, 10, 480, 200);

    /* Draw background lines */
    color.red = 0;
    color.green = 20000;
    color.blue = 0;
    gdk_gc_set_rgb_fg_color(gc, &color);
    for (int i = 10; i <= 230; i += 20)
        gdk_draw_line(swap_curve->window, gc, 15, i, 495, i);
    for (int i = 15; i <= 480; i += 20)
        gdk_draw_line(swap_curve->window, gc, i + mem_curve_start, 10, i + mem_curve_start, 210);

    /* Settle memory curve start position to make it live */
    swap_curve_start -= 4;
    if (swap_curve_start == 0)
        swap_curve_start = 20;

    /* Initial data */
    if (flag == 0) {
        for (int i = 0; i < 120; i++) {
            swap_ratio_data[i] = 0;
            flag = 1;
        }
    }

    /* Add data */
    swap_ratio_data[now_pos] = swap_ratio / 100;
    now_pos++;
    if (now_pos == 120)
        now_pos = 0;
```

```
    /* Draw lines */
    color.red = 0;
                color.green = 65535;
                color.blue = 0;
                gdk_gc_set_rgb_fg_color(gc, &color);
    draw_pos = now_pos;
    for (int i = 0; i < 119; i++) {
        gdk_draw_line(swap_curve->window, gc,
                              15 + i * 4, 210 - 200 * swap_ratio_data[draw_pos % 120],
                              15 + (i + 1) * 4, 210 - 200 * swap_ratio_data[(draw_pos + 1) % 120]);
        draw_pos++;
        if (draw_pos == 120)
            draw_pos = 0;
    }

    /* Reset the color */
    color.red = 0;
                color.green = 0;
                color.blue = 0;
                gdk_gc_set_rgb_fg_color(gc, &color);

    /* To loop this function, it must return TRUE */
    return TRUE;
}

/*
  * get_cpu_ratio - Get CPU use ratio from /proc/stat
  */
gboolean get_cpu_ratio(gpointer label) {
    /*
      * stat file format
      *
      * cpu user nice system idle iowait
      *
      * t1, t2: Two near moments
      * cpu(total) = user+nice+system+idle
      * pcpu = 100 *(total – idle) / total
      * total =total(t2) – total(t1)
      * idle =idle(t2) – idle(t1)
      */
    /* Data holder */
    static long old_idle, old_total;
    static int flag = 0;

    long user, nice, system, idle, total;
    float total_diff, idle_diff;
    char cpu[10], buffer[256], cpu_ratio_char[256];
    int fd;
    fd = open("/proc/stat", O_RDONLY);
    read(fd, buffer, sizeof(buffer));
    close(fd);
    sscanf(buffer, "%s %ld %ld %ld %ld", cpu, &user, &nice, &system, &idle);

    /* First */
    if (flag == 0) {
        flag = 1;
```

```
        old_idle = idle;
        old_total = user + nice + system + idle;
        cpu_ratio = 0;
    }
    /* Others */
    else {
        total = user + nice + system + idle;
        total_diff = total - old_total;
        idle_diff = idle - old_idle;
        cpu_ratio = 100 * (total_diff - idle_diff) / total_diff;
        total = old_total;
        idle = old_idle;
    }
    sprintf(cpu_ratio_char, "CPU usage: %0.1f%%", cpu_ratio);
    gtk_label_set_text(GTK_LABEL(label), cpu_ratio_char);
    return TRUE;
}

/*
  * get_cpu_mhz - Get CPU freqency from /proc/cpuinfo line 8
  */
gboolean get_cpu_mhz(gpointer label) {
    int fd;
    char info_buf[1024];
    char info_str[1024];
    char cpu_freq_char[256];
    char *pos = NULL;
    int i;
    fd = open("/proc/cpuinfo", O_RDONLY);
    read(fd, info_buf, sizeof(info_buf));
    close(fd);

    /* Read CPU mhz */
    i = 0;
    pos = strstr(info_buf, "cpu MHz");
    while (*pos != ':')
        pos++;
    pos += 2;

    while (*pos != '\n') {
        info_str[i] = *pos;
        i++;
        pos++;
    }
    info_str[i] = '\0';
    strcpy(cpu_freq, info_str);
    sprintf(cpu_freq_char, "CPU frequency: %s MHz", cpu_freq);
    gtk_label_set_text(GTK_LABEL(label), cpu_freq_char);
    return TRUE;
}

/*
  * get_memory_ratio - Get memory use ratio in /proc/meminfo
  */
gboolean get_memory_ratio(gpointer label) {
    int fd;
    char mem_buf[1024];
```

```
    char mem_total_char[1024];
    char mem_free_char[1024];
    char mem_ratio_char[1024];
    char *pos = NULL;
    int i;
    fd = open("/proc/meminfo", O_RDONLY);
    read(fd, mem_buf, sizeof(mem_buf));
    close(fd);

    /* Read memory total */
    i = 0;
    pos = strstr(mem_buf, "MemTotal");
    while (*pos != ':')
        pos++;
    pos += 1;
    while (*pos == ' ')
        pos++;

    while (*pos != ' ') {
        mem_total_char[i] = *pos;
        i++;
        pos++;
    }
    mem_total_char[i] = '\0';
    mem_total = atof(mem_total_char) / (1024 * 1024);

    /* Read memory free */
    i = 0;
    pos = strstr(mem_buf, "MemFree");
    while (*pos != ':')
        pos++;
    pos += 1;
    while (*pos == ' ')
        pos++;

    while (*pos != ' ') {
        mem_free_char[i] = *pos;
        i++;
        pos++;
    }
    mem_free_char[i] = '\0';
    mem_free = atof(mem_free_char) / (1024 * 1024);

    /* Get memory use ratio */
    mem_ratio = 100 - (mem_free / mem_total) * 100;
    sprintf(mem_ratio_char, "Memory usage: %0.1f%%", mem_ratio);
    gtk_label_set_text(GTK_LABEL(label), mem_ratio_char);
    return TRUE;
}

/*
  * get_memory_fraction - Set memory use fraction
  */
gboolean get_memory_fraction(gpointer label) {
    char mem_fraction[1024];
    sprintf(mem_fraction, "%0.2f / %0.2f GB", mem_total - mem_free, mem_total);
    gtk_label_set_text(GTK_LABEL(label), mem_fraction);
```

```
    return TRUE;
}

/*
 * get_swap_ratio - Get sawp use ratio in /proc/meminfo
 */
gboolean get_swap_ratio(gpointer label) {
    int fd;
    char swap_buf[1024];
    char swap_total_char[1024];
    char swap_free_char[1024];
    char swap_ratio_char[1024];
    char *pos = NULL;
    int i;
    fd = open("/proc/meminfo", O_RDONLY);
    read(fd, swap_buf, sizeof(swap_buf));
    close(fd);

    /* Read swap total */
    i = 0;
    pos = strstr(swap_buf, "SwapTotal");
    while (*pos != ':')
        pos++;
    pos += 1;
    while (*pos == ' ')
        pos++;

    while (*pos != ' ') {
        swap_total_char[i] = *pos;
        i++;
        pos++;
    }
    swap_total_char[i] = '\0';
    swap_total = atof(swap_total_char) / (1024 * 1024);

    /* Read swap free */
    i = 0;
    pos = strstr(swap_buf, "SwapFree");
    while (*pos != ':')
        pos++;
    pos += 1;
    while (*pos == ' ')
        pos++;

    while (*pos != ' ') {
        swap_free_char[i] = *pos;
        i++;
        pos++;
    }
    swap_free_char[i] = '\0';
    swap_free = atof(swap_free_char) / (1024 * 1024);

    /* Get swap use ratio */
    swap_ratio = 100 - (swap_free / swap_total) * 100;
    sprintf(swap_ratio_char, "Swap usage: %0.1f%%", swap_ratio);
    gtk_label_set_text(GTK_LABEL(label), swap_ratio_char);
    return TRUE;
```

```
}

/*
  * get_swap_fraction - Set swap use fraction
  */
gboolean get_swap_fraction(gpointer label) {
    char swap_fraction[1024];
    sprintf(swap_fraction, "%0.2f / %0.2f GB", swap_total - swap_free, swap_total);
    gtk_label_set_text(GTK_LABEL(label), swap_fraction);
    return TRUE;
}

/*
  * get_sys_info - Set system information
  */
gboolean get_sys_info(gpointer label) {
    int fd;
    FILE *fp;
    int i, j;
    char buffer[1024];
    char *pos = NULL;
    char host_name[128];
    char os_name[128];
    int os_type;
    char kernel_version[128];
    char gcc_version[128];
    int setup_time;
    int uphour, upminute, upsecond;

    /* Get host name */
    fp = fopen("/etc/hostname", "r");
    fgets(host_name, sizeof(host_name), fp);
    fclose(fp);

    /* Get os name */
    memset(buffer, 0, sizeof(buffer));
    fd = open("/etc/issue", O_RDONLY);
    read(fd, buffer, sizeof(buffer));
    close(fd);
    for (i = 0; i < 1024; i++) {
        if (buffer[i] == '\\')
            break;
    }
    buffer[i - 1] = '\0';
    strcpy(os_name, buffer);

    /* Get os type */
    os_type = sizeof(char *) * 8;

    /* Get kernel version */
    fp = fopen("/proc/sys/kernel/osrelease", "r");
    fgets(kernel_version, sizeof(kernel_version), fp);
    fclose(fp);

    /* Get gcc version */
    memset(buffer, 0, sizeof(buffer));
    fd = open("/proc/version", O_RDONLY);
```

```
    read(fd, buffer, sizeof(buffer));
    close(fd);
    pos = strstr(buffer, "gcc version");
    for (i = 0, j = 0; i < 1024; i++) {
        if (*pos == ' ')
            j++;
        if (j == 2)
            break;
        pos++;
    }
    pos++;
    for (i = 0; i < 1024; i++) {
        if (pos[i] == ' ')
            break;
        gcc_version[i] = pos[i];
    }
    gcc_version[i] = '\0';

    /* Get setup time */
    memset(buffer, 0, sizeof(buffer));
    fd = open("/proc/uptime", O_RDONLY);
    read(fd, buffer, sizeof(buffer));
    close(fd);
    for (i = 0; i < 1024; i++) {
        if (buffer[i] == ' ')
            break;
    }
    buffer[i] = '\0';
    setup_time = atoi(buffer);
    upsecond = setup_time % 60;
    upminute = (setup_time / 60) % 60;
    uphour = setup_time / 3600;

    sprintf(buffer, "Hostname:                %s\n\
OS Name:                  %s\n\n\
OS Type:                    %d-bit\n\n\
Kernel Version:       %s\n\
GCC Version:            %s\n\n\
Uptime:                     %02d:%02d:%02d",
                host_name, os_name, os_type, kernel_version, gcc_version, uphour, upminute, upsecond);

    gtk_label_set_text(GTK_LABEL(label), buffer);
    set_label_fontsize((GtkWidget *)label, "13");

    return TRUE;
}

/*
 * get_network_info - Get network information in /proc/net/dev
 */
gboolean get_network_info(gpointer label) {
    /*
     * Network dev file format:
     *
     * Inter-|   Receive                                                |  Transmit
     *  face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo
colls carrier compressed
```

```
     *      dev1: ...
     *      dev2: ...
     *      ...
     */
    FILE *fp;
    long r_byte, s_byte;
    long receive_byte, send_byte;
    float receive_diff, send_diff;
    static long old_receive, old_send;
    static int flag = 0;
    char buffer[256];
    char *pos;

    fp = fopen("/proc/net/dev", "r");
    fgets(buffer, sizeof(buffer), fp);
    fgets(buffer, sizeof(buffer), fp);

    receive_byte = 0;
    send_byte = 0;

    while ((pos = fgets(buffer, sizeof(buffer), fp)) != NULL) {
        while (*pos != ':')
            pos++;
        pos++;
        sscanf(pos, "%ld %*d %*d %*d %*d %*d %*d %*d %ld", &r_byte, &s_byte);
        receive_byte += r_byte;
        send_byte += s_byte;
    }
    fclose(fp);

    if (flag == 0) {
        old_receive = receive_byte;
        old_send = send_byte;
        receive_diff = 0;
        send_diff = 0;
        flag = 1;
    }
    else {
        receive_diff = receive_byte - old_receive;
        send_diff = send_byte - old_send;
        old_receive = receive_byte;
        old_send = send_byte;
    }

    /* Count speed (KB) */
    receive_speed = receive_diff / (1024 * 2);
    send_speed = send_diff / (1024 * 2);

    sprintf(buffer, "        Network Speed:\n\n\
\tUpload speed: %7.1f KB/s\tDownload speed: %7.1f KB/s\n\n\
\t    Total upload: %7.1f MB\t\t    Total download: %7.1f MB",
                send_speed, receive_speed, send_byte / (1024.0 * 1024.0), receive_byte / (1024.0 * 1024.0));
    gtk_label_set_text(GTK_LABEL(label), buffer);
    set_label_fontsize((GtkWidget *)label, "13");

    return TRUE;
}
```

```
/*
  * get_disk_info - Get network information in /proc/diskstat
  */
gboolean get_disk_info(gpointer label) {
    /*
     * Disk status file format:
     *
     * 8 num sdx rd_ios rd_merges rd_sectors rd_ticks wr_ios wr_merges wr_sectors \
     *    wr_ticks in_flight io_ticks time_in_queue
     *
     * read speed = (Δrd_sectors/Δt) * (block_size / 1024)
     * write speed = (Δwr_sectors/Δt) * (block_size / 1024)
     * Here block_size is 512
     */
    FILE *fp;
    long rd_sector, wr_sector;
    long rd_sectors, wr_sectors;
    float rd_diff, wr_diff;
    static long old_rd_sectors, old_wr_sectors;
    static int flag = 0;
    char buffer[256];
    char *pos;

    fp = fopen("/proc/diskstats", "r");

    rd_sectors = 0;
    wr_sectors = 0;

    while((pos = fgets(buffer, sizeof(buffer), fp)) != NULL) {
        sscanf(buffer, "%*d %*d %*s %*d %*d %d %*d %*d %*d %d", &rd_sector, &wr_sector);
        rd_sectors += rd_sector;
        wr_sectors += wr_sector;
    }
    fclose(fp);

    if (flag == 0) {
        old_rd_sectors = rd_sectors;
        old_wr_sectors = wr_sectors;
        rd_diff = 0;
        wr_diff = 0;
        flag = 1;
    }
    else {
        rd_diff = rd_sectors - old_rd_sectors;
        wr_diff = wr_sectors - old_wr_sectors;
        old_rd_sectors = rd_sectors;
        old_wr_sectors = wr_sectors;
    }

    /* Count speed (MB) */
    read_speed = (rd_diff * 512) / (2 * 1024 * 1024);
    write_speed = (wr_diff * 512) / (2 * 1024 * 1024);

    sprintf(buffer, "Disk I/O Speed:\n\n\
\tRead speed: %7.1f MB/s\t\tWrite speed: %7.1f MB/s\n\n\
\t    Total Read: %7.1f GB\t\t    Total Write: %7.1f GB",
```

```
                read_speed, write_speed, (rd_sectors * 512) / (1024.0 * 1024.0 * 1024.0), (wr_sectors * 512) /
(1024.0 * 1024.0 * 1024.0));
    gtk_label_set_text(GTK_LABEL(label), buffer);
    set_label_fontsize((GtkWidget *)label, "13");

    return TRUE;
}

/*******************ASSISTS*******************/


/*
  * utf8_fix - To settle waring: Invalid UTF-8 string passed to pango_layout_set_text()
  *
  * Referencing from:
https://stackoverflow.com/questions/43753260/pango-warning-invalid-utf-8-string-passed-to-pango-layout-set-text-in-gtk
  */
char* utf8_fix(char *c) {
    return g_locale_to_utf8(c, -1, NULL, NULL, NULL);
}

/*
  * scroll_to_line - To set the scroll window position
  *
  * Modified from: https://my.oschina.net/plumsoft/blog/79950
  */
void scroll_to_line(gpointer scrolled_window, gint line_num, gint to_line_index) {
    GtkAdjustment *adj;
    gdouble lower_value, upper_value, page_size, max_value, line_height, to_value;
    adj = gtk_scrolled_window_get_vadjustment(GTK_SCROLLED_WINDOW(scrolled_window));
    lower_value = gtk_adjustment_get_lower(adj);
    upper_value = gtk_adjustment_get_upper(adj);
    page_size = gtk_adjustment_get_page_size(adj);
    max_value = upper_value - page_size;
    line_height = upper_value / line_num;
    to_value = line_height * to_line_index;
    if (to_value < lower_value)
        to_value = lower_value;
    if (to_value > max_value)
        to_value = max_value;
    gtk_adjustment_set_value(adj, to_value);
    return;
}

/*
  * set_label_fontsize - To set font size in a label
  *
  * Referencing from: https://blog.csdn.net/gl_ding/article/details/4939355
  */
void set_label_fontsize(GtkWidget *label, char *fontsize) {
    PangoFontDescription *desc_info = pango_font_description_from_string(fontsize);
    gtk_widget_modify_font(label, desc_info);
    pango_font_description_free(desc_info);
}
```

设计内容五 模拟文件系统

filesystem.h – 文件系统库头文件

```
/*
 * HUST OS Design - Part V
 *
 * File system simulation - Header
 *
 * Created by zxcpyp at 2018-08-26
 *
 * Github: zxc479773533
 */

#ifndef ZXCPYP_FILESYSTEM
#define ZXCPYP_FILESYSTEM

#include "../lib/zxcpyp_sys.h"
#include "fs_error.h"
#include <time.h>
#include <wait.h>

/* Disk parameters */
#define BLOCKSIZE 1024                  /* The block size */
#define INODENUM 1024                   /* The Inode nums */
#define BLOCKNUM (1024 * 64)            /* The disk size 64 MB */
#define FILEBLKMAX (1024 * 4)           /* The file max size is 4 MB */
#define FILENAMEMAX 20                  /* File and directory name max length */
#define USERNAMEMAX 20                  /* User name max length */
#define USERPWDMAX 20                   /* User password max length */
#define USERNUMMAX 10

#define INODESIZE sizeof(inode)                                     /* The inode size */
#define SUPERPOS sizeof(sys_users)                                  /* The #1 is super block */
#define INODEPOS (SUPERPOS + sizeof(super_block))                   /* The #2 ~ #1025 are inodes */
#define BLOCKPOS (INODEPOS + INODESIZE * INODENUM)                  /* The #1026 is first free block */


#define TYPE_DIR 0
#define TYPE_FILE 1

#define DIRMAXINBLK (BLOCKSIZE / sizeof(directory)) /* Directory entry num per block */
#define DIRMAXNUM (FILEBLKMAX * DIRMAXINBLK)                /* File num per directory */

#define CAN_READ 0
#define CAN_WRITE 1

#define BUFFERFILE "/tmp/zxcpyp_disk_buf"

/* File system data structures */
typedef struct super_block {
    int inode_map[INODENUM];
    int block_map[BLOCKNUM];
    int inode_free_num;
    int block_free_num;
} super_block;
```

```
typedef struct inode {
    int block_used[FILEBLKMAX];
    int block_used_num;
    int size;
    int mode;
    time_t creat_time;
    time_t modify_time;
    int user_id;
} inode;

typedef struct directory {
    char name[FILENAMEMAX];
    int inode_id;
} directory;

typedef struct user {
    char user_name[USERNAMEMAX];
    char user_pwd[USERPWDMAX];
} user;

typedef struct sys_users {
    int user_map[USERNUMMAX];
    int user_num;
    user users[USERNUMMAX];
} sys_users;

/* Global variables */
extern FILE *disk;
extern super_block super;
extern int current_inode_id;
extern inode current_inode;
extern directory current_dir_content[DIRMAXNUM];
extern int current_dir_num;
extern int current_user_id;

extern char path[128];

/* Developer functions */
void reset_disk();
void print_current_user_id(void);
void print_current_inode_id(void);
void print_current_dir_num(void);
void show_files_info();
void print_superblk_inode_info(int pos);
void print_superblk_block_info(int pos);
void show_users_info(void);

/* Core function */
int inode_alloc(void);
int inode_free(int ino);
int init_dir_inode(int new_ino, int ino);
int init_file_inode(int new_ino);
int block_alloc(void);
int block_free(int bno);

/* User function */
int login(void);
```

```
int user_pwd(void);
int user_add(char *name, char *pwd);
int user_del(char *name);

/* Disk function */
int load_super_block(void);
int format_disk(void);
int close_disk(void);

/* File and directory function */
int dir_open(int ino);
int dir_close(int ino);
int dir_creat(int ino, int type, char *name);
int dir_rm(int ino, int type, char *name);
int dir_cd(int ino, char *path);
int dir_ls(void);
int dir_ls_l(void);
int file_open(int ino, char *name);
int file_close(int ino, char *name);
int file_cat(void);
int file_mv(int ino, char *srcname, char *dstname);
int file_cp(int ino, char *srcname, char *dstname);

/* Assist function */
int oct2dec(int oct_number);
int check_name(char *name);
int check_type(int ino, int type);
int check_mode(int mode, int operation);
void path_change(int old_inode_id, char *name);
int mtime_change(int ino, char *name);
void get_modestr(char *modstr, int mode);
int mode_change(int mode, char *name);
int check_if_readonly(int ino, char *name);

#endif // !ZXCPYP_FILESYSTEM
```

filesystem.c – 文件系统库实现

```
/*
 * HUST OS Design - Part V
 *
 * File system simulation - Implementation
 *
 * Created by zxcpyp at 2018-08-26
 *
 * Github: zxc479773533
 */

#include "filesystem.h"

FILE *disk;
super_block super;
int current_inode_id;
inode current_inode;
directory current_dir_content[DIRMAXNUM];
```

```
int current_dir_num;
int current_user_id;
char path[128];


/*********************DEVELOPER FUNCTIONS*********************/
/*
  * Warning: These functions is only for developer, they will not
  *                be called in the formal edition.
  */


/*
  * reset_disk - Totally reset the disk, you can call it to reset
  *                      disk or make a new file become a disk file for this
  *                      program.
  *
  * root initial password: 123456
  */
void reset_disk(void) {
    sys_users all_users;
    /* Reset user */
    fseek(disk, 0, SEEK_SET);
    fread(&all_users, sizeof(sys_users), 1, disk);
    memset(&all_users, 0, sizeof(sys_users));

    /* Set root initial password */
    all_users.user_map[0] = 1;
    all_users.user_num = 1;
    strcpy(all_users.users[0].user_name, "root");
    strcpy(all_users.users[0].user_pwd, "123456");

    /* Save data */
    fseek(disk, 0, SEEK_SET);
    fwrite(&all_users, sizeof(sys_users), 1, disk);

    /* Format the disk */
    format_disk();
}

/*
  * print_current_user_id - See the current user id
  */
void print_current_user_id(void) {
    printf("%d\n", current_user_id);
}


/*
  * print_current_inode_id - See the current inode id
  */
void print_current_inode_id(void) {
    printf("%d\n", current_inode_id);
}

/*
  * print_current_inode_id - See the current directory num
```

```
 */
void print_current_dir_num(void) {
    printf("%d\n", current_dir_num);
}

/*
 * show_files_info - See the files inode information in directory
 */
void show_files_info(void) {
    int pos;
    inode node;
    for (pos = 0; pos < current_dir_num; pos++) {
        fseek(disk, INODEPOS + current_dir_content[pos].inode_id * INODESIZE, SEEK_SET);
        fread(&node, sizeof(node), 1, disk);
        printf("pos: %d    ", pos);
        printf("name: %-10s    ", current_dir_content[pos].name);
        printf("inode id: %d    ", current_dir_content[pos].inode_id);
        printf("user id: %d\n\n", node.user_id);
    }
}

/*
 * print_superblk_inode_info - See inode information in super block
 */
void print_superblk_inode_info(int pos) {
    printf("Super block inode:\n");
    printf("Pos %d: %d\n", pos, super.inode_map[pos]);
    printf("Free num: %d\n", super.inode_free_num);
}

/*
 * print_superblk_block_info - See block information in super block
 */
void print_superblk_block_info(int pos) {
    printf("Super block block:\n");
    printf("Pos %d: %d\n", pos, super.block_map[pos]);
    printf("Free num: %d\n", super.block_free_num);
}

void show_users_info(void) {
    int pos;
    sys_users all_users;

    /* Read users inforation */
    fseek(disk, 0, SEEK_SET);
    fread(&all_users, sizeof(sys_users), 1, disk);

    for (pos = 0; pos < USERNUMMAX; pos++) {
        if (all_users.user_map[pos] == 1) {
            printf("User id: %d\n", pos);
            printf("User id: %s\n\n", all_users.users[pos].user_name);
        }
    }
}

/********************CORE FUNCTIONS********************/
```

```
/*
  * inode_alloc - Alloc a new free inode from inode map
  */
int inode_alloc(void) {
    int ino;
    if (super.inode_free_num <= 0)
        return FS_NO_INODE;
    super.inode_free_num--;
    for (ino = 0; ino < INODENUM; ino++) {
        if (super.inode_map[ino] == 0) {
            super.inode_map[ino] = 1;
            break;
        }
    }
    return ino;
}

/*
  * inode_free - Free a inode and the file's all block
  */
int inode_free(int ino) {
    inode node;
    fseek(disk, INODEPOS + ino * INODESIZE, SEEK_SET);
    fread(&node, sizeof(inode), 1, disk);
    for (int i = 0; i < node.block_used_num; i++)
        block_free(node.block_used[i]);
    super.inode_map[ino] = 0;
    super.inode_free_num++;
    return FS_OK;
}

/*
  * init_dir_inode - Initial directory inode
  */
int init_dir_inode(int new_ino, int ino) {
    int bno;
    inode node;
    directory basic_link[2];

    fseek(disk, INODEPOS + new_ino * INODESIZE, SEEK_SET);
    fread(&node, sizeof(inode), 1, disk);
    bno = block_alloc();

    /* Set new inode information */
    node.block_used[0] = bno;
    node.block_used_num = 1;
    node.size = 2 * sizeof(directory);
    node.mode = oct2dec(1755);
    time_t timer;
                time(&timer);
    node.creat_time = timer;
    node.modify_time = timer;
    node.user_id = current_user_id;

    /* Save inode information */
    fseek(disk, INODEPOS + new_ino * INODESIZE, SEEK_SET);
    fwrite(&node, sizeof(inode), 1, disk);
```

```
    /* Set basic links */
    strcpy(basic_link[0].name, ".");
    basic_link[0].inode_id = new_ino;
    strcpy(basic_link[1].name, "..");
    basic_link[1].inode_id = ino;

    /* Save basic links */
    fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
    fwrite(basic_link, sizeof(directory), 2, disk);

    return FS_OK;
}

/*
  * init_file_inode - Initial file inode
  */
int init_file_inode(int new_ino) {
    inode node;
    fseek(disk, INODEPOS + new_ino * INODESIZE, SEEK_SET);
    fread(&node, sizeof(inode), 1, disk);

    /* Set new inode information */
    node.block_used_num = 0;
    node.size = 0;
    node.mode = oct2dec(644);
    time_t timer;
                time(&timer);
    node.creat_time = timer;
    node.modify_time = timer;
    node.user_id = current_user_id;

    /* Save inode information */
    fseek(disk, INODEPOS + new_ino * INODESIZE, SEEK_SET);
    fwrite(&node, sizeof(inode), 1, disk);

    return FS_OK;
}

/*
  * block_alloc - Alloc a new free block from block map
  */
int block_alloc(void) {
    int bno;
    if (super.block_free_num <= 0)
        return FS_NO_BLOCK;
    super.block_free_num--;
    for (bno = 0; bno < BLOCKNUM; bno++) {
        if (super.block_map[bno] == 0) {
            super.block_map[bno] = 1;
            break;
        }
    }
    return bno;
}

/*
```

```
 * block_free - Free a block
 */
int block_free(int bno) {
    super.block_free_num++;
    super.block_map[bno] = 0;
    return FS_OK;
}


/*********************USER FUNCTIONS*********************/


/*
 * login - Login for the disk
 */
int login(void) {
    sys_users all_users;
    char uname[USERNAMEMAX];
    char upwd[USERPWDMAX];
    printf("localhost login: ");
    scanf("%s", uname);
    printf("\nPassword: ");
    scanf("%s", upwd);
    getchar();

    /* Read users inforation */
    fseek(disk, 0, SEEK_SET);
    fread(&all_users, sizeof(sys_users), 1, disk);

    /* Check user */
    for (int i = 0; i < USERNUMMAX; i++) {
        if (strcmp(all_users.users[i].user_name, uname) == 0) {
            if (strcmp(all_users.users[i].user_pwd, upwd) == 0) {
                current_user_id = i;
                sprintf(path, "%s@localhost: / >", uname);
                printf("\033[2J");
                return FS_LOGIN;
            }
        }
    }

    usleep(500000);
    printf("Login incorrect\n\n");
    return FS_LOGIN_ERROR;
}

/*
 * user_pwd - Change a user's password
 */
int user_pwd(void) {
    int i;
    sys_users all_users;
    char current_pwd[USERPWDMAX], new_pwd[USERPWDMAX], new_pwd_2[USERPWDMAX];

    /* Read users inforation */
    fseek(disk, 0, SEEK_SET);
    fread(&all_users, sizeof(sys_users), 1, disk);
```

```
    printf("Change password for %s\n", all_users.users[current_user_id].user_name);
    printf("Current password: ");
    scanf("%s", current_pwd);
    getchar();

    if (strcmp(all_users.users[current_user_id].user_pwd, current_pwd) != 0) {
        printf("passwd: Identification failure\n");
        printf("passwd: Password not changed\n");
        return FS_INVALID;
    }

    printf("New password: ");
    scanf("%s", new_pwd);
    printf("New password again: ");
    scanf("%s", new_pwd_2);
    getchar();

    if (strcmp(new_pwd, new_pwd_2) != 0) {
        printf("Sorry, the password does not match\n");
        printf("passwd: Password service preliminary check failed\n");
        printf("passwd: Password not changed\n");
        return FS_INVALID;
    }

    strcpy(all_users.users[current_user_id].user_pwd, new_pwd);
    printf("password: Password successfully updated\n");

    /* Save users inforation */
    fseek(disk, 0, SEEK_SET);
    fwrite(&all_users, sizeof(sys_users), 1, disk);

    return FS_OK;
}

/*
  * user_add - Add a user
  */
int user_add(char *name, char *pwd) {
    sys_users all_users;

    /* Read users inforation */
    fseek(disk, 0, SEEK_SET);
    fread(&all_users, sizeof(sys_users), 1, disk);

    /* Check user */
    for (int i = 0; i < USERNUMMAX; i++) {
        if (strcmp(all_users.users[i].user_name, name) == 0) {
            return FS_USER_EXIST;
        }
    }

    for (int i = 0; i < USERNUMMAX; i++) {
        if (all_users.user_map[i] == 0) {
            all_users.user_map[i] = 1;
            strcpy(all_users.users[i].user_name, name);
            strcpy(all_users.users[i].user_pwd, pwd);
```

```
            break;
        }
    }

    /* Save users inforation */
    fseek(disk, 0, SEEK_SET);
    fwrite(&all_users, sizeof(sys_users), 1, disk);

    return FS_OK;
}


/*
  * user_del - Delete a user
  */
int user_del(char *name) {
    sys_users all_users;
    int i;

    /* Read users inforation */
    fseek(disk, 0, SEEK_SET);
    fread(&all_users, sizeof(sys_users), 1, disk);

    /* Check user */
    for (i = 0; i < USERNUMMAX; i++) {
        if (strcmp(all_users.users[i].user_name, name) == 0) {
            memset(all_users.users[i].user_name, 0, USERNAMEMAX);
            memset(all_users.users[i].user_pwd, 0, USERPWDMAX);
            all_users.user_map[i] = 0;
            break;
        }
    }
    if (i == USERNUMMAX)
        return FS_USER_NOT_EXIST;

    /* Save users inforation */
    fseek(disk, 0, SEEK_SET);
    fwrite(&all_users, sizeof(sys_users), 1, disk);

    return FS_OK;
}


/*********************DISK FUNCTIONS*********************/


/*
  * load_super_block - Load super block information from disk
  */
int load_super_block(void) {
    int ret;

    /* Load super block */
    fseek(disk, SUPERPOS, SEEK_SET);
    ret = fread(&super, sizeof(super_block), 1, disk);
    if (ret != 1)
        return FS_RD_ERROR;
```

```
    /* Open root dir directory */
    current_inode_id = 0;
    ret = dir_open(current_inode_id);
    if (ret != FS_OK)
        return ret;

    return FS_OK;
}

/*
  * format_disk - Format and initialize the disk
  */
int format_disk(void) {
    int ret;

    /* Set inode map */
    memset(super.inode_map, 0, sizeof(super.inode_map));
    super.inode_map[0] = 1;
    super.inode_free_num = INODENUM - 1;

    /* Set block map */
    memset(super.block_map, 0, sizeof(super.block_map));
    super.block_map[0] = 1;
    super.block_free_num = BLOCKNUM - 1;

    /* Set root inode */
    current_inode_id = 0;
    fseek(disk, INODEPOS, SEEK_SET);
    ret = fread(&current_inode, sizeof(inode), 1, disk);
    if (ret != 1)
        return FS_RD_ERROR;
    current_inode.block_used[0] = 0;
    current_inode.block_used_num = 1;
    current_inode.size = 2 * sizeof(directory);
    current_inode.mode = oct2dec(1755);
    time_t timer;
                time(&timer);
    current_inode.creat_time = timer;
    current_inode.modify_time = timer;
    current_inode.user_id = 0;

    /* Set basic link of root file */
    current_dir_num = 2;
    strcpy(current_dir_content[0].name, ".");
    current_dir_content[0].inode_id = 0;
    strcpy(current_dir_content[1].name, "..");
    current_dir_content[1].inode_id = 0;

    strcpy(path, "root@localhost: / >");

    return FS_OK;
}


/*
  * close_disk - Load super block information from disk
```

```
 */
int close_disk(void) {
    int ret;

    /* Save super block */
    fseek(disk, SUPERPOS, SEEK_SET);
    ret = fwrite(&super, sizeof(super_block), 1, disk);
    if (ret != 1)
        return FS_WR_ERROR;

    /* Close current directory */
    ret = dir_close(current_inode_id);
    if (ret != 1)
        return ret;

    return FS_OK;
}


/*********************DIR FUNCTIONS*********************/


/*
 * dir_open - Open a directory, and read its data in memory
 */
int dir_open(int ino) {
    int i, ret;
    int end_block_dirnum;
    fseek(disk, INODEPOS + ino * INODESIZE, SEEK_SET);
    ret = fread(&current_inode, sizeof(inode), 1, disk);
    if (ret != 1)
        return FS_RD_ERROR;

    /* Read all directory entry */
    for (i = 0; i < current_inode.block_used_num - 1; i++) {
        fseek(disk, BLOCKPOS + current_inode.block_used[i] * BLOCKSIZE, SEEK_SET);
        fread(current_dir_content + i * DIRMAXINBLK, sizeof(directory), DIRMAXINBLK, disk);
    }
    end_block_dirnum = current_inode.size / sizeof(directory) - DIRMAXINBLK *
(current_inode.block_used_num - 1);
    fseek(disk, BLOCKPOS + current_inode.block_used[i] * BLOCKSIZE, SEEK_SET);
    fread(current_dir_content + i * DIRMAXINBLK, sizeof(directory), end_block_dirnum, disk);

    time_t timer;
    time(&timer);
    current_inode.modify_time = timer;
    current_dir_num = i * DIRMAXINBLK + end_block_dirnum;
    return FS_OK;
}

/*
 * dir_close - Close a directory, and write its data back to disk
 */
int dir_close(int ino) {
    int i, ret;
    int end_block_dirnum;
```

```
    /* Write all directory entry back to disk */
    for (i = 0; i < current_inode.block_used_num - 1; i++) {
        fseek(disk, BLOCKPOS + current_inode.block_used[i] * BLOCKSIZE, SEEK_SET);
        fwrite(current_dir_content + i * DIRMAXINBLK, sizeof(directory), DIRMAXINBLK, disk);
    }
    end_block_dirnum = current_dir_num - i * DIRMAXINBLK;
    fseek(disk, BLOCKPOS + current_inode.block_used[i] * BLOCKSIZE, SEEK_SET);
    fwrite(current_dir_content + i * DIRMAXINBLK, sizeof(directory), end_block_dirnum, disk);

    current_inode.size = current_dir_num * sizeof(directory);
    fseek(disk, INODEPOS + ino * INODESIZE, SEEK_SET);
    ret = fwrite(&current_inode, sizeof(inode), 1, disk);
    if (ret != 1)
        return FS_WR_ERROR;
    return FS_OK;
}

/*
  * dir_creat - Creat a directory or a file and initial the inode
  */
int dir_creat(int ino, int type, char *name) {
    int new_ino;
    int block_need = 1;
    if (current_dir_num >= DIRMAXNUM)
        return FS_DIR_FULL;
    if (check_name(name) != FS_OK)
        return FS_FILE_EXIST;

    /* Check mode */
    if (check_mode(ino, CAN_WRITE) == 0)
        return FS_NO_PRIVILAGE;

    /* If need more blocks */
    if (current_dir_num / DIRMAXINBLK != (current_dir_num + 1) / DIRMAXINBLK)
        block_need++;
    if (block_need > super.block_free_num)
        return FS_NO_BLOCK;
    if (block_need == 2)
        current_inode.block_used[++current_inode.block_used_num] = block_alloc();
    new_ino = inode_alloc();
    if (new_ino == FS_NO_INODE)
        return FS_NO_INODE;

    /* Initial new inode */
    if (type == TYPE_DIR)
        init_dir_inode(new_ino, ino);
    else
        init_file_inode(new_ino);

    /* Register new inode */
    strcpy(current_dir_content[current_dir_num].name, name);
    current_dir_content[current_dir_num].inode_id = new_ino;

    /* Update modify time */
    time_t timer;
            time(&timer);
    current_inode.modify_time = timer;
```

```
    current_dir_num++;
    return FS_OK;
}

/*
  * dir_rm - Delete a empty directory or a file
  */
int dir_rm(int ino, int type, char *name) {
    int rm_inode;
    int ret;
    inode node;

    /* Check mode */
    if (check_mode(ino, CAN_WRITE) == 0)
        return FS_NO_PRIVILAGE;

    /* Can't delete . and .. */
    if (!strcmp(name, ".") || !strcmp(name, ".."))
        return FS_INVALID;

    /* Check if the directory or file exists */
    for (rm_inode = 0; rm_inode < current_dir_num; rm_inode++) {
        if (strcmp(name, current_dir_content[rm_inode].name) == 0)
            break;
    }
    if (rm_inode == current_dir_num)
        return FS_NO_EXIST;

    rm_inode = current_dir_content[rm_inode].inode_id;
    /* Read inode information */
    fseek(disk, INODEPOS + rm_inode * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);

    /* Check user */
    if (node.user_id != current_user_id && current_user_id != 0)
        return FS_NO_PRIVILAGE;

    /* Check type */
    ret = check_type(node.mode, type);
    if (ret == FS_ISNOT_DIR || ret == FS_ISNOT_FILE)
        return ret;

    if (ret == FS_IS_DIR) {
        dir_cd(ino, name);
        if (current_dir_num != 2) {
            dir_cd(rm_inode, "..");
            return FS_DIR_NOEMPTY;
        }
        dir_cd(rm_inode, "..");
    }

    int pos;
    inode_free(rm_inode);
    for (pos = 0; pos < current_dir_num; pos++) {
        if (strcmp(current_dir_content[pos].name, name) == 0)
            break;
```

```
    }
    for (; pos < current_dir_num - 1; pos++) {
        current_dir_content[pos] = current_dir_content[pos + 1];
    }
    current_dir_num--;

    /* Free last block if need */
    if (current_dir_num / DIRMAXINBLK != (current_dir_num - 1) / DIRMAXINBLK) {
        current_inode.block_used_num--;
        block_free(current_inode.block_used[current_inode.block_used_num]);
    }

    /* Update modify time */
    time_t timer;
                time(&timer);
    current_inode.modify_time = timer;

    return FS_OK;
}

/*
  * dir_cd - Enter into a directory
  */
int dir_cd(int ino, char *name) {
    int i;
    int cd_inode;
    inode node;

    /* Check if the directory or file exists */
    for (i = 0; i < current_dir_num; i++) {
        if (strcmp(current_dir_content[i].name, name) == 0)
            break;
    }
    if (i == current_dir_num)
        return FS_NO_EXIST;
    cd_inode = current_dir_content[i].inode_id;

    /* Check if this is a directory */
    fseek(disk, INODEPOS + cd_inode * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);
    if (check_type(node.mode, TYPE_FILE) == FS_IS_FILE)
        return FS_ISNOT_DIR;

    /* Check mode */
    if (check_mode(cd_inode, CAN_READ) == 0)
        return FS_NO_PRIVILAGE;

    dir_close(ino);
    current_inode_id = cd_inode;
    dir_open(cd_inode);

    return FS_OK;
}

/*
  * dir_ls - List all files in directory
  */
```

```
int dir_ls(void) {
    /* Save current status */
    dir_close(current_inode_id);
    dir_open(current_inode_id);

    int pos;
    inode node;
    for (pos = 0; pos < current_dir_num; pos++) {
        fseek(disk, INODEPOS + current_dir_content[pos].inode_id * INODESIZE, SEEK_SET);
        fread(&node, sizeof(node), 1, disk);
        if (check_type(node.mode, TYPE_DIR) == FS_IS_DIR)
            printf("\e[1;34m%s\t\e[0m", current_dir_content[pos].name);
        else
            printf("%s\t", current_dir_content[pos].name);
    }
    printf("\n");

    return FS_OK;
}

/*
  * dir_ls_l - List all files and its information in a directory
  */
int dir_ls_l(void) {
    /* Save current status */
    dir_close(current_inode_id);
    dir_open(current_inode_id);

    int pos;
    sys_users all_users;
    inode node;
    char modstr[11];
    char *time;

    /* Read users inforation */
    fseek(disk, 0, SEEK_SET);
    fread(&all_users, sizeof(sys_users), 1, disk);

    for (pos = 0; pos < current_dir_num; pos++) {
        fseek(disk, INODEPOS + current_dir_content[pos].inode_id * INODESIZE, SEEK_SET);
        fread(&node, sizeof(node), 1, disk);

        get_modestr(modstr, node.mode);
        time = ctime(&node.modify_time);

        if (check_type(node.mode, TYPE_DIR) == FS_IS_DIR) {
            printf("%s    %-s \e[1;34m%6ld\e[0m %.12s \e[1;34m%-s\e[0m\n",
                    modstr, all_users.users[node.user_id].user_name, node.size / sizeof(directory), time + 4,
current_dir_content[pos].name);
        }
        else {
            printf("%s    %-s %6d %.12s %-s\n",
                    modstr, all_users.users[node.user_id].user_name, node.size, time + 4,
current_dir_content[pos].name);
        }
    }
```

```
    return FS_OK;
}

/*
  * file_open - Open a file, read its contains into buffer tmp file
  */
int file_open(int ino, char *name) {
    int open_inode;
    int bno, pos;
    inode node;
    char block[BLOCKSIZE];
    FILE *buf_fp = fopen(BUFFERFILE, "w+");

    /* Check if the directory or file exists */
    for (open_inode = 0; open_inode < current_dir_num; open_inode++) {
        if (strcmp(current_dir_content[open_inode].name, name) == 0)
            break;
    }
    if (open_inode == current_dir_num)
        return FS_NO_EXIST;

    open_inode = current_dir_content[open_inode].inode_id;

    /* Check mode */
    if (check_mode(open_inode, CAN_READ) == 0)
        return FS_NO_PRIVILAGE;

    /* Read inode information */
    fseek(disk, INODEPOS + open_inode * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);

    /* Check type */
    if (check_type(node.mode, TYPE_DIR) == FS_IS_DIR)
        return FS_IS_DIR;

    if (node.size == 0) {
        fclose(buf_fp);
        return FS_OK;
    }

    /* Read data from disk */
    for (pos = 0; pos < node.block_used_num - 1; pos++) {
        memset(block, 0, BLOCKSIZE);
        bno = node.block_used[pos];
        fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
        fread(block, sizeof(char), BLOCKSIZE, disk);
        fwrite(block, sizeof(char), BLOCKSIZE, buf_fp);
        block_free(bno);
        node.size -= BLOCKSIZE;
    }
    bno = node.block_used[pos];
    fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
    fread(block, sizeof(char), node.size, disk);
    fwrite(block, sizeof(char), node.size, buf_fp);
    block_free(bno);
    node.size = 0;
    node.block_used_num = 0;
```

```
    /* Save inode */
    fseek(disk, INODEPOS + open_inode * INODESIZE, SEEK_SET);
    fwrite(&node, sizeof(inode), 1, disk);

    fclose(buf_fp);
    return FS_OK;
}

/*
  * file_close - Close a file, write buffer file contents to disk
  */
int file_close(int ino, char *name) {
    int close_inode;
    int bno, read_num;
    inode node;
    char block[BLOCKSIZE];
    FILE *buf_fp = fopen(BUFFERFILE, "r");

    /* Check if the directory or file exists */
    for (close_inode = 0; close_inode < current_dir_num; close_inode++) {
        if (strcmp(current_dir_content[close_inode].name, name) == 0)
            break;
    }
    if (close_inode == current_dir_num)
        return FS_NO_EXIST;

    close_inode = current_dir_content[close_inode].inode_id;

    /* Read inode information */
    fseek(disk, INODEPOS + close_inode * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);

    /* Check type */
    if (check_type(node.mode, TYPE_DIR) == FS_IS_DIR)
        return FS_ISNOT_FILE;

    /* Read data from buffer file */
    memset(block, 0, BLOCKSIZE);
    read_num = fread(block, sizeof(char), BLOCKSIZE, buf_fp);
    while(read_num != 0) {
        bno = block_alloc();
        if (bno == FS_NO_BLOCK)
            break;
        fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
        fwrite(block, sizeof(char), BLOCKSIZE, disk);
        node.block_used[node.block_used_num] = bno;
        node.block_used_num++;
        node.size += read_num;

        memset(block, 0, BLOCKSIZE);
        read_num = fread(block, sizeof(char), BLOCKSIZE, buf_fp);
    }

    /* Save inode */
    fseek(disk, INODEPOS + close_inode * INODESIZE, SEEK_SET);
    fwrite(&node, sizeof(inode), 1, disk);
```

```
    fclose(buf_fp);
    return FS_OK;
}

/*
  * file_cat - Get a file's content
  */
int file_cat(void) {
    int read_num;
    FILE *buf_fp = fopen(BUFFERFILE, "r");
    char block[BLOCKSIZE];
    memset(block, 0, BLOCKSIZE);

    /* Read data from buffer file */
    while((read_num = fread(block, sizeof(char), BLOCKSIZE, buf_fp) != 0))
        printf("%s", block);

    fclose(buf_fp);
    return FS_OK;
}

/*
  * file_mv - Move file from srcname to dstname
  */
int file_mv(int ino, char *srcname, char *dstname) {
    int pos;
    int src_inode;
    inode node;

    /* Check if the directory or file exists */
    for (pos = 0; pos < current_dir_num; pos++) {
        if (strcmp(current_dir_content[pos].name, srcname) == 0)
            break;
    }
    if (pos == current_dir_num)
        return FS_NO_EXIST;

    src_inode = current_dir_content[pos].inode_id;

    /* Check mode */
    if (check_mode(current_inode_id, CAN_READ) == 0)
        return FS_NO_PRIVILAGE;

    /* Read inode information */
    fseek(disk, INODEPOS + src_inode * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);

    /* Check type */
    if (check_type(node.mode, TYPE_DIR) == FS_IS_DIR)
        return FS_IS_DIR;

    /* Move to other directory */
    if (dstname[strlen(dstname) - 1] == '/') {
        dstname[strlen(dstname) - 1] = '\0';
        int dstpos, dst_node;
        inode dstnode;
```

```
/* Check if the directory or file exists */
for (dstpos = 0; dstpos < current_dir_num; dstpos++) {
   if (strcmp(current_dir_content[dstpos].name, dstname) == 0)
      break;
}
if (dstpos == current_dir_num)
   return FS_NO_EXIST;

dst_node = current_dir_content[dstpos].inode_id;

/* Check mode */
if (check_mode(dst_node, CAN_WRITE) == 0)
   return FS_NO_PRIVILAGE;

/* Read inode information */
fseek(disk, INODEPOS + dst_node * INODESIZE, SEEK_SET);
fread(&dstnode, sizeof(dstnode), 1, disk);

/* Check type */
if (check_type(dstnode.mode, TYPE_FILE) == FS_IS_FILE)
   return FS_IS_FILE;

/* Check same name */
dir_cd(current_inode_id, dstname);
if (check_name(srcname) == FS_FILE_EXIST) {
   dir_close(current_inode_id);
   current_inode_id = ino;
   dir_open(ino);
   return FS_FILE_EXIST;
}
dir_close(current_inode_id);
current_inode_id = ino;
dir_open(ino);

/* Delete entry */
for (; pos < current_dir_num - 1; pos++) {
   current_dir_content[pos] = current_dir_content[pos + 1];
}
current_dir_num--;

/* Free last block if need */
if (current_dir_num / DIRMAXINBLK != (current_dir_num - 1) / DIRMAXINBLK) {
   current_inode.block_used_num--;
   block_free(current_inode.block_used[current_inode.block_used_num]);
}

/* Create new entry */
dir_cd(current_inode_id, dstname);
dir_creat(current_inode_id, TYPE_FILE, srcname);

/* Copy inode */
for (dstpos = 0; dstpos < current_dir_num; dstpos++) {
   if (strcmp(current_dir_content[dstpos].name, srcname) == 0)
      break;
}
dst_node = current_dir_content[dstpos].inode_id;
```

```
        fseek(disk, INODEPOS + dst_node * INODESIZE, SEEK_SET);
        fwrite(&node, sizeof(node), 1, disk);

        /* Return to src directory */
        dir_close(current_inode_id);
        current_inode_id = ino;
        dir_open(ino);
    }
    /* Move to current directory */
    else {
        /* Check mode */
        if (check_mode(current_inode_id, CAN_WRITE) == 0)
            return FS_NO_PRIVILAGE;

        /* Check same name */
        if (check_name(dstname) == FS_FILE_EXIST)
            return FS_FILE_EXIST;

        strcpy(current_dir_content[pos].name, dstname);
        /* Save data */
        dir_close(current_inode_id);
        dir_open(ino);
    }

    return FS_OK;
}

/*
  * file_cp - Copy file from srcname to dstname
  */
int file_cp(int ino, char *srcname, char *dstname) {
    int pos, bno;
    int src_inode;
    inode node;
    char block[BLOCKSIZE];
    FILE *buf_fp = fopen(BUFFERFILE, "w+");

    /* Check if the directory or file exists */
    for (pos = 0; pos < current_dir_num; pos++) {
        if (strcmp(current_dir_content[pos].name, srcname) == 0)
            break;
    }
    if (pos == current_dir_num)
        return FS_NO_EXIST;

    src_inode = current_dir_content[pos].inode_id;

    /* Check mode */
    if (check_mode(current_inode_id, CAN_READ) == 0)
        return FS_NO_PRIVILAGE;

    /* Read inode information */
    fseek(disk, INODEPOS + src_inode * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);

    /* Check type */
    if (check_type(node.mode, TYPE_DIR) == FS_IS_DIR)
```

```
    return FS_IS_DIR;

/* Copy file content */
if (node.size == 0) {
    fclose(buf_fp);
    return FS_OK;
}
for (pos = 0; pos < node.block_used_num - 1; pos++) {
    memset(block, 0, BLOCKSIZE);
    bno = node.block_used[pos];
    fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
    fread(block, sizeof(char), BLOCKSIZE, disk);
    fwrite(block, sizeof(char), BLOCKSIZE, buf_fp);
}
bno = node.block_used[pos];
fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
fread(block, sizeof(char), node.size, disk);
fwrite(block, sizeof(char), node.size, buf_fp);
fclose(buf_fp);

/* Copy to other directory */
if (dstname[strlen(dstname) - 1] == '/') {
    dstname[strlen(dstname) - 1] = '\0';
    int dstpos, dst_node;
    inode dstnode;

    /* Check if the directory or file exists */
    for (dstpos = 0; dstpos < current_dir_num; dstpos++) {
        if (strcmp(current_dir_content[dstpos].name, dstname) == 0)
            break;
    }
    if (dstpos == current_dir_num)
        return FS_NO_EXIST;

    dst_node = current_dir_content[dstpos].inode_id;

    /* Check mode */
    if (check_mode(dst_node, CAN_WRITE) == 0)
        return FS_NO_PRIVILAGE;

    /* Read inode information */
    fseek(disk, INODEPOS + dst_node * INODESIZE, SEEK_SET);
    fread(&dstnode, sizeof(dstnode), 1, disk);

    /* Check type */
    if (check_type(dstnode.mode, TYPE_FILE) == FS_IS_FILE)
        return FS_IS_FILE;

    /* Check same name */
    dir_cd(current_inode_id, dstname);
    if (check_name(srcname) == FS_FILE_EXIST) {
        dir_close(current_inode_id);
        current_inode_id = ino;
        dir_open(ino);
        return FS_FILE_EXIST;
    }
    dir_close(current_inode_id);
```

```
        current_inode_id = ino;
        dir_open(ino);

        /* Create new entry */
        dir_cd(current_inode_id, dstname);
        dir_creat(current_inode_id, TYPE_FILE, srcname);

        /* Save file content */
        file_close(current_inode_id, srcname);

        /* Return to src directory */
        dir_close(current_inode_id);
        current_inode_id = ino;
        dir_open(ino);
    }
    /* Copy to current directory */
    else {
        /* Check mode */
        if (check_mode(current_inode_id, CAN_WRITE) == 0)
            return FS_NO_PRIVILAGE;

        /* Check same name */
        if (check_name(dstname) == FS_FILE_EXIST)
            return FS_FILE_EXIST;

        /* Create new file */
        dir_creat(ino, TYPE_FILE, dstname);

        /* Save file content */
        file_close(ino, dstname);
    }

    return FS_OK;
}


/********************ASSIST FUNCTIONS********************/


/*
 * oct2dec - Change Octal number to decimal number
 *
 * For this program use, only supply 0~7777
 */
int oct2dec(int oct_number) {
    int dec_number = 0;
    int a, b, c, d;
    a = oct_number / 1000;
    b = (oct_number % 1000) / 100;
    c = (oct_number % 100) / 10;
    d = oct_number % 10;
    dec_number = ((a * 8 + b) * 8 + c) * 8 + d;
    return dec_number;
}

/*
 * check_name - Check if the file already exists
```

```
 */
int check_name(char *name) {
    int i;
    for (i = 0; i < current_dir_num; i++) {
        if (strcmp(name, current_dir_content[i].name) == 0)
            return FS_FILE_EXIST;
    }
    return FS_OK;
}

/*
 * check_type - Check file type
 */
int check_type(int mode, int type) {
    int isdir = mode & (1 << 9);
    if (isdir == (1 << 9) && type == TYPE_FILE)
        return FS_ISNOT_FILE;
    else if (isdir == 0 && type == TYPE_DIR)
        return FS_ISNOT_DIR;
    else if (isdir == 0 && type == TYPE_FILE)
        return FS_IS_FILE;
    else
        return FS_IS_DIR;
}

/*
 * check_mode - Check if user have privilege to do operation
 */
int check_mode(int ino, int operation) {
    inode node;
    int ret;

    fseek(disk, INODEPOS + ino * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);

    if (operation == CAN_READ) {
        if (node.user_id == current_user_id)
            ret = ((node.mode & (1 << 8)) == (1 << 8));
        else
            ret = ((node.mode & (1 << 2)) == (1 << 2));
    }
    else {
        if (node.user_id == current_user_id)
            ret = ((node.mode & (1 << 7)) == (1 << 7));
        else
            ret = ((node.mode & (1 << 1)) == (1 << 1));
    }
    return ret;
}

/*
 * path_cd - Change path when enter a directory
 */
void path_change(int old_inode_id, char *name) {
    int pos;
    if (!strcmp(name, ".") || (!strcmp(name, "..") && (old_inode_id == 0)))
        return;
```

```
    else if (!strcmp(name, "..") && current_inode_id != 0) {
        for (pos = strlen(path) - 1; pos >= 0; pos--) {
            if (path[pos] == '/') {
                path[pos] = '\0';
                strcat(path, " >");
                break;
            }
        }
    }
    else if (!strcmp(name, "..") && current_inode_id == 0) {
        for (pos = strlen(path) - 1; pos >= 0; pos--) {
            if (path[pos] == '/') {
                path[pos + 1] = '\0';
                strcat(path, " >");
                break;
            }
        }
    }
    else if (path[strlen(path) - 3] == '/') {
        path[strlen(path) - 2] = '\0';
        strcat(path, name);
        strcat(path, " >");
    }
    else {
        path[strlen(path) - 2] = '\0';
        strcat(path, "/");
        strcat(path, name);
        strcat(path, " >");
    }
}

/*
  * mtime_change - Change modified time for a file
  */
int mtime_change(int ino, char *name) {
    int i;
    int ch_node;
    inode node;

    /* Check if the directory or file exists */
    for (i = 0; i < current_dir_num; i++) {
        if (strcmp(current_dir_content[i].name, name) == 0)
            break;
    }
    ch_node = current_dir_content[i].inode_id;

    /* Check if this is a directory */
    fseek(disk, INODEPOS + ch_node * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);

    /* Update modify time */
    time_t timer;
                time(&timer);
    node.modify_time = timer;

    /* Save node data */
    fseek(disk, INODEPOS + ch_node * INODESIZE, SEEK_SET);
```

```
    fwrite(&node, sizeof(node), 1, disk);

    return FS_OK;
}

/*
  * path_cd - Change path when enter a directory
  */
void get_modestr(char *modstr, int mode) {
    strcpy(modstr, "----------");
    if ((mode & (1 << 9)) == (1 << 9))
        modstr[0] = 'd';
    if ((mode & (1 << 8)) == (1 << 8))
        modstr[1] = 'r';
    if ((mode & (1 << 7)) == (1 << 7))
        modstr[2] = 'w';
    if ((mode & (1 << 6)) == (1 << 6))
        modstr[3] = 'x';
    if ((mode & (1 << 5)) == (1 << 5))
        modstr[4] = 'r';
    if ((mode & (1 << 4)) == (1 << 4))
        modstr[5] = 'w';
    if ((mode & (1 << 3)) == (1 << 3))
        modstr[6] = 'x';
    if ((mode & (1 << 2)) == (1 << 2))
        modstr[7] = 'r';
    if ((mode & (1 << 1)) == (1 << 1))
        modstr[8] = 'w';
    if ((mode & 1) == 1)
        modstr[9] = 'x';
}

/*
  * mode_change - Change a file's mode
  */
int mode_change(int mode, char *name) {
    int i;
    int change_inode;
    inode node;

    /* Invalid mode */
    if (mode < 0 || mode > 777)
        return FS_INVALID_MODE;

    /* Check if the directory or file exists */
    for (i = 0; i < current_dir_num; i++) {
        if (strcmp(current_dir_content[i].name, name) == 0)
            break;
    }
    if (i == current_dir_num)
        return FS_NO_EXIST;
    change_inode = current_dir_content[i].inode_id;

    /* Check if this is a directory */
    fseek(disk, INODEPOS + change_inode * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);
```

```
    /* Only owner can change the mode */
    if (node.user_id != current_user_id)
        return FS_NO_PRIVILAGE;

    mode = oct2dec(mode);
    mode |= (node.mode & (1 << 9));

    node.mode = mode;

    /* Save node data */
    fseek(disk, INODEPOS + change_inode * INODESIZE, SEEK_SET);
    fwrite(&node, sizeof(node), 1, disk);

    return FS_OK;
}

int check_if_readonly(int ino, char *name) {
    int pid, status;
    int check_inode;
    int bno, pos;
    inode node;
    char block[BLOCKSIZE];
                char *vim_arg[] = {"vim", BUFFERFILE, NULL};
    FILE *buf_fp = fopen(BUFFERFILE, "w+");

    /* Check if the directory or file exists */
    for (check_inode = 0; check_inode < current_dir_num; check_inode++) {
        if (strcmp(current_dir_content[check_inode].name, name) == 0)
            break;
    }
    if (check_inode == current_dir_num)
        return FALSE;

    check_inode = current_dir_content[check_inode].inode_id;

    /* Check mode */
    if (check_mode(check_inode, CAN_READ) == 0 || check_mode(check_inode, CAN_WRITE) == 1)
        return FALSE;

    /* Read only */

    /* Read inode information */
    fseek(disk, INODEPOS + check_inode * INODESIZE, SEEK_SET);
    fread(&node, sizeof(node), 1, disk);

    /* Check type */
    if (check_type(node.mode, TYPE_DIR) == FS_IS_DIR)
        return FALSE;

    /* Read data from disk */
    for (pos = 0; pos < node.block_used_num - 1; pos++) {
        memset(block, 0, BLOCKSIZE);
        bno = node.block_used[pos];
        fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
        fread(block, sizeof(char), BLOCKSIZE, disk);
        fwrite(block, sizeof(char), BLOCKSIZE, buf_fp);
    }
```

```
    bno = node.block_used[pos];
    fseek(disk, BLOCKPOS + bno * BLOCKSIZE, SEEK_SET);
    fread(block, sizeof(char), node.size, disk);
    fwrite(block, sizeof(char), node.size, buf_fp);

    fclose(buf_fp);

    if ((pid = fork()) == 0) {
        execvp("vim", vim_arg);
   }

    wait(&status);
    return TRUE;
}
```

fs_error.h – 错误宏定义

```
#ifndef ZXCPYP_FS_ERROR
#define ZXCPYP_FS_ERROR

#define FS_OK 0
#define FS_NO_EXIST -1
#define FS_RD_ERROR -2
#define FS_WR_ERROR -3
#define FS_NO_BLOCK -4
#define FS_NO_INODE -5
#define FS_DIR_FULL -6
#define FS_FILE_EXIST -7
#define FS_NO_PRIVILAGE -8
#define FS_ISNOT_FILE -9
#define FS_IS_FILE -10
#define FS_ISNOT_DIR -11
#define FS_IS_DIR -12
#define FS_DIR_NOEMPTY -13
#define FS_USER_EXIST -14
#define FS_USER_NOT_EXIST -15
#define FS_INVALID_MODE -16

#define FS_INVALID -98

#define FS_LOGIN -99
#define FS_LOGIN_ERROR -100

#endif // !ZXCPYP_FS_ERROR
```

zxcpyp_exec.h – 功能操作实现

```
/*
 * zxcpyp exec - Execute function for zxcpyp prompt
 *
 * Created by zxcpyp at 2018-02-22
 *
 * Github: zxc479773533
```

```
 *
 * Used for: HUST OS Design - Part V in 2018-08-26
 */

#include "filesystem.h"

/* Decide whether use developer APIs */
int developer = 1;

/*
 * print_help - Print help messages
 */
void print_help(void) {
    printf("ZXCPYP File System version 1.2\n\n");
    printf("Supported cmd:\n");
    printf("    Directory and File operations:\n");
    printf("        %-8s %-8s %-8s %-8s %-8s\n", "mkdir", "rmdir", "cd", "ls [-l]", "touch");
    printf("        %-8s %-8s %-8s %-8s\n", "vim", "cat", "cp", "mv");
    printf("    User operations:\n");
    printf("        %-8s %-8s %-8s\n", "useradd", "userdel", "passwd");
    printf("    Other operations:\n");
    printf("        %-8s %-8s %-8s %-8s\n", "fmt", "chmod", "help", "exit");
    if (developer == 0) {
        printf("    Developer operations:\n");
        printf("        %-8s %-8s %-8s %-8s %-8s\n", "reset", "puid", "pino", "dirnum", "show");
        printf("        %-8s %-8s %-8s\n", "users", "superi", "superb");
    }
}

/*
 * builtin_cmd - Judge buildin command
 */
int developer_cmd(int argc, char **argv) {
    if (developer == 1)
        return 0;
    if (!strcmp(argv[0], "reset")) {
        reset_disk();
        return 1;
    }
    if (!strcmp(argv[0], "puid")) {
        print_current_user_id();
        return 1;
    }
    if (!strcmp(argv[0], "pino")) {
        print_current_inode_id();
        return 1;
    }
    if (!strcmp(argv[0], "dirnum")) {
        print_current_dir_num();
        return 1;
    }
    if (!strcmp(argv[0], "show")) {
        show_files_info();
        return 1;
    }
    if (!strcmp(argv[0], "users")) {
        show_users_info();
```

```
        return 1;
    }
    if (argc == 2 && !strcmp(argv[0], "superi")) {
        print_superblk_inode_info(atoi(argv[1]));
        return 1;
    }
    if (argc == 2 && !strcmp(argv[0], "superb")) {
        print_superblk_block_info(atoi(argv[1]));
        return 1;
    }
    /* Not a developer command */
    return 0;
}

/*
  * developer_cmd - Judge developer command
  */
int builtin_cmd(char **argv) {
    if (!strcmp(argv[0], "fmt")) {
        if (current_user_id == 0)
            format_disk();
        else
            printf("fmt: You need root privilege!\n");
        return 1;
    }
    if (!strcmp(argv[0], "passwd")) {
        user_pwd();
        return 1;
    }
    if (!strcmp(argv[0], "exit")) {
        printf("[EXIT] User-exit. Terminated!\n");
        close_disk();
        fclose(disk);
        exit(0);
    }
    if (!strcmp(argv[0], "help")) {
        print_help();
        return 1;
    }
    /* Not a built-in command */
    return 0;
}

/*
  * py_execute - The main execute function
  */
int py_execute(char *func , int argc, char **argv) {
    int ret;
    if (!strcmp(func, "mkdir")) {
        if (argc != 2) {
            printf("Usage: mkdir [dirname]\n");
            return 1;
        }
        ret = dir_creat(current_inode_id, TYPE_DIR, argv[1]);
        if (ret == FS_FILE_EXIST)
            printf("mkdir: Fail to create directory \"%s\": File already exists\n", argv[1]);
        else if (ret == FS_NO_PRIVILAGE)
```

```
            printf("mkdir: Fail to create directory \"%s\": Insufficient privilege\n", argv[1]);
        else if (ret != FS_OK)
            printf("mkdir: Fail to create directory \"%s\": No enough space\n", argv[1]);
        return 1;
    }
    if (!strcmp(func, "rmdir")) {
        if (argc != 2) {
            printf("Usage: rmdir [dirname]\n");
            return 1;
        }
        ret = dir_rm(current_inode_id, TYPE_DIR, argv[1]);
        if (ret == FS_INVALID)
            printf("rmdir: Fail to delete \"%s\": Invalid operation\n", argv[1]);
        else if (ret == FS_NO_PRIVILAGE)
            printf("rmdir: Fail to delete \"%s\": Insufficient privilege\n", argv[1]);
        else if (ret == FS_NO_EXIST)
            printf("rmdir: Fail to delete \"%s\": File not exists\n", argv[1]);
        else if (ret == FS_NO_PRIVILAGE)
            printf("rmdir: Fail to delete \"%s\": Insufficient privilege\n", argv[1]);
        else if (ret == FS_ISNOT_DIR)
            printf("rmdir: Fail to delete \"%s\": Not a directory\n", argv[1]);
        else if (ret == FS_DIR_NOEMPTY)
            printf("rmdir: Fail to delete \"%s\": Directory not empty\n", argv[1]);
        return 1;
    }
    if (!strcmp(func, "cd")) {
        if (argc != 2) {
            printf("Usage: cd [dirname]\n");
            return 1;
        }
        int old_inode_id = current_inode_id;
        ret = dir_cd(current_inode_id, argv[1]);
        if (ret == FS_NO_EXIST)
            printf("cd: No such file or directory: \"%s\"\n", argv[1]);
        else if (ret == FS_NO_PRIVILAGE)
            printf("cd: Insufficient privilege: \"%s\"\n", argv[1]);
        else if (ret == FS_ISNOT_DIR)
            printf("cd: Not a directory: \"%s\"\n", argv[1]);
        else if (ret == FS_OK)
            path_change(old_inode_id, argv[1]);
        return 1;
    }
    if (!strcmp(func, "ls")) {
        if (argc == 2 && !strcmp(argv[1], "-l"))
            dir_ls_l();
        else
            dir_ls();
        return 1;
    }
    if (!strcmp(func, "touch")) {
        if (argc != 2) {
            printf("Usage: touch [filename]\n");
            return 1;
        }
        ret = dir_creat(current_inode_id, TYPE_FILE, argv[1]);
        if (ret == FS_FILE_EXIST)
            mtime_change(current_inode_id, argv[1]);
```

```
    else if (ret == FS_NO_PRIVILAGE)
        printf("touch: Fail to create file \"%s\": Insufficient privilege\n", argv[1]);
    else if (ret != FS_OK)
        printf("touch: Fail to create file \"%s\": No enough space\n", argv[1]);
    return 1;
}
if (!strcmp(func, "rm")) {
    if (argc != 2) {
        printf("Usage: rm [filename]\n");
        return 1;
    }
    ret = dir_rm(current_inode_id, TYPE_FILE, argv[1]);
    if (ret == FS_INVALID)
        printf("rmdir: Fail to delete \"%s\": Invalid operation\n", argv[1]);
    else if (ret == FS_NO_EXIST)
        printf("rmdir: Fail to delete \"%s\": File not exists\n", argv[1]);
    else if (ret == FS_NO_PRIVILAGE)
        printf("rmdir: Fail to delete \"%s\": Insufficient privilege\n", argv[1]);
    else if (ret == FS_ISNOT_FILE)
        printf("rmdir: Fail to delete \"%s\": Not a file\n", argv[1]);
    return 1;
}
if (!strcmp(func, "vim")) {
    if (argc != 2) {
        printf("Usage: vim [filename]\n");
        return 1;
    }
    int pid, status;
            char *vim_arg[] = {"vim", BUFFERFILE, NULL};
    if (check_if_readonly(current_inode_id, argv[1]) == TRUE) {
        printf("vim: Fail to save \"%s\": Insufficient privilege\n", argv[1]);
        return 1;
    }
    ret = file_open(current_inode_id, argv[1]);
    if (ret == FS_IS_DIR) {
        printf("vim: Fail to open \"%s\": Is a directory\n", argv[1]);
        return 1;
    }
    else if (ret == FS_NO_PRIVILAGE) {
        printf("vim: Fail to open \"%s\": Insufficient privilege\n", argv[1]);
        return 1;
    }
    else if (ret == FS_NO_EXIST) {
        ret = dir_creat(current_inode_id, TYPE_FILE, argv[1]);
        if (ret == FS_NO_PRIVILAGE) {
            printf("vim: Fail to creat \"%s\": Insufficient privilege\n", argv[1]);
            return 1;
        }
        file_open(current_inode_id, argv[1]);
    }
    if((pid = fork()) == 0) {
            execvp("vim", vim_arg);
    }
    wait(&status);
    file_close(current_inode_id, argv[1]);
    return 1;
}
```

```
    if (!strcmp(func, "cat")) {
        if (argc != 2) {
            printf("Usage: cat [filename]\n");
            return 1;
        }
        ret = file_open(current_inode_id, argv[1]);
        if (ret == FS_IS_DIR)
            printf("cat: \"%s\": Is a directory\n", argv[1]);
        else if (ret == FS_NO_EXIST)
            printf("cat: \"%s\": No such file or directory\n", argv[1]);
        else if (ret == FS_NO_PRIVILAGE)
            printf("cat: \"%s\": Insufficient privilege\n", argv[1]);
        else {
            file_cat();
            file_close(current_inode_id, argv[1]);
        }
        return 1;
    }
    if (!strcmp(func, "useradd")) {
        if (argc != 3) {
            printf("Usage: useradd [username] [userpwd]\n");
            return 1;
        }
        if (current_user_id != 0)
            printf("useradd: You need root privilege!\n");
        else {
            ret = user_add(argv[1], argv[2]);
            if (ret == FS_USER_EXIST)
                printf("useradd: Failed to add user \"%s\": User already exists\n", argv[1]);
        }
        return 1;
    }
    if (!strcmp(func, "userdel")) {
        if (argc != 2) {
            printf("Usage: userdel [username]\n");
            return 1;
        }
        if (current_user_id != 0)
            printf("userdel: You need root privilege!\n");
        else {
            ret = user_del(argv[1]);
            if (ret == FS_USER_NOT_EXIST)
                printf("userdel: Failed to delete user \"%s\": User not exists\n", argv[1]);
        }
        return 1;
    }
    if (!strcmp(func, "chmod")) {
        if (argc != 3) {
            printf("Usage: userdel [mod] [filename]\n");
            return 1;
        }
        ret = mode_change(atoi(argv[1]), argv[2]);
        if (ret == FS_NO_PRIVILAGE)
            printf("chmod: Change the permissions of \"%s\": Invalid operation\n", argv[2]);
        else if (ret == FS_NO_EXIST)
            printf("chmod: Unable to access \"%s\": No such file or directory\n", argv[2]);
        else if (ret == FS_INVALID_MODE)
```

```
            printf("chmod: Invalid operation: \"%s\"\n", argv[1]);
        return 1;
    }
    if (!strcmp(func, "mv")) {
        if (argc != 3) {
            printf("Usage: mv [srcfile] [dstfile | dstdir/]\n");
            return 1;
        }
        ret = file_mv(current_inode_id, argv[1], argv[2]);
        if (ret == FS_NO_EXIST)
            printf("mv: Unable to get file status for \"%s\" or \"%s\" (stat) : No such file or directory\n",
argv[1], argv[2]);
        else if (ret == FS_NO_PRIVILAGE)
            printf("mv: Unable to move \"%s\": Insufficient privilege\n", argv[1]);
        else if (ret == FS_IS_DIR)
            printf("mv: Unable to move \"%s\": Not a file\n", argv[1]);
        else if (ret == FS_IS_FILE)
            printf("mv: Unable to move \"%s\" into \"%s\": Not a directory\n", argv[1], argv[2]);
        else if (ret == FS_FILE_EXIST)
            printf("mv: Unable to move \"%s\": Target file exists\n", argv[1]);
        return 1;
    }
    if (!strcmp(func, "cp")) {
        if (argc != 3) {
            printf("Usage: cp [srcfile] [dstfile | dstdir/]\n");
            return 1;
        }
        ret = file_cp(current_inode_id, argv[1], argv[2]);
        if (ret == FS_NO_EXIST)
            printf("cp: Unable to get file status for \"%s\" or \"%s\" (stat) : No such file or directory\n", argv[1],
argv[2]);
        else if (ret == FS_NO_PRIVILAGE)
            printf("cp: Unable to copy \"%s\": Insufficient privilege\n", argv[1]);
        else if (ret == FS_IS_DIR)
            printf("cp: Unable to copy \"%s\": Not a file\n", argv[1]);
        else if (ret == FS_IS_FILE)
            printf("cp: Unable to copy \"%s\" into \"%s\": Not a directory\n", argv[1], argv[2]);
        else if (ret == FS_FILE_EXIST)
            printf("cp: Unable to move \"%s\": Target file exists\n", argv[1]);
        return 1;
    }
    return 0;
}
```

zxcpyp_prompt.h – shell 框架实现

```
/*
 * zxcpyp prompt - A interactive shell template
 *
 * Created by zxcpyp at 2018-02-22
 *
 * Github: zxc479773533
 *
 * Used for: HUST OS Design - Part V in 2018-08-26
 */
```

```
#include "zxcpyp_exec.h"

/* Constants */
#define MAXLINE 1024
#define MAXARGS 128

/* Global variables */

// char prompt[] = "zxcpyp > ";
char theme[] = "Here to control zxcpyp's file system";

/* Founction prototypes */
void eval(char *cmdline);
int parseline(const char *cmdline, char **argv);

/* Print file system version */
void print_version() {
    char load[] = ".....................";
    char ch[20];
    printf("ZXCPYP File System: version v1.2\n\n");
    printf("Copyright (C) 2018 zxcpyp\n\n");
    for (int i = 0; i < 19; i++) {
        memset(ch, 0, sizeof(ch));
        strncpy(ch, load, i + 1);
        printf("loading %s\n", ch);
        fflush(stdout);
        usleep(100000);
        printf("\033[1A\033[K");
    }
    printf("loading .....................\n\n");
}

/* Print usage messages */
void print_usage(void) {
    printf("Usage: ./zxcpypfs [options]\n");
    printf("Options:\n");
    printf("\t-h: print this massges\n");
    printf("\t-p: hide the prompt\n");
    printf("\t-d: use developer mode\n");
    printf("\tdefault: start shell\n");
}

int Start_Shell(int argc, char **argv) {
    int ret;
    char ch;
    /* The command line */
    char cmdline[MAXLINE];
    /* Decide whether print a prompt, default yes*/
    int emit_prompt = 1;

    /* Parse the command line */
    while ((ch = getopt(argc, argv, "hpd")) != EOF) {
        switch(ch) {
        case 'h':
            print_usage();
            exit(0);
```

```
        case 'p':
            emit_prompt = 0;
            break;
        case 'd':
            developer = 0;
            break;
        default:
            print_usage();
            exit(1);
        }
    }

    /* Login */
    print_version();
    while (login() != FS_LOGIN) {}
    ret = load_super_block();
    if (ret == FS_RD_ERROR)
        return FS_RD_ERROR;

    /* Print informations */
    printf("Wherecome to zxcpyp's interactive shell!\n");
    printf("%s\n", theme);
    printf("\n");

    /* Excute the shell's read/eval loop */
    while (1) {

        /* Print prompt */
        if (emit_prompt) {
            if (developer == 0)
                printf("\e[1;31m[Developer mode] \e[0m");
            printf("\e[1;32m%s \e[0m", path);
            fflush(stdout);
        }
        if ((fgets(cmdline, MAXLINE, stdin) == NULL) && ferror(stdin)) {
            printf("fgets error\n");
            exit(1);
        }
        if (feof(stdin)) {
            printf("[EXIT] Parser reached end-of-file. Terminated!\n");
            fflush(stdout);
            exit(0);
        }

        /* Evaluate the command line */
        eval(cmdline);
        fflush(stdout);
    }

    return 0;
}

/*
 * eval - Evaluate the command line that the user has just typed in
 *
 * If the user has requseted a built-in command, then excute it immediately.
 */
```

```
void eval(char *cmdline) {
    char *argv[MAXARGS];
    char buf[MAXLINE];
    int argc;

    /* Parse command */
    strcpy(buf, cmdline);
    argc = parseline(buf, argv);

    if (argv[0] == NULL)
        return;
    if (!developer_cmd(argc, argv)) {
        if (!builtin_cmd(argv)) {
            if (!py_execute(argv[0], argc, argv))
                printf("zxcpypsh: command not found: %s\n", argv[0]);
            // else
                // printf("[INFO] Operation finished.\n\n");
        }
    }
}

/*
  * parseline - Parse the command line and build the argv array.
  */
int parseline(const char *cmdline, char **argv) {
    /* Holds local copy of command line */
    static char array[MAXLINE];
    char *buf = array;
    char *delim;
    int argc;

    strcpy(buf, cmdline);
    buf[strlen(buf) - 1] = ' ';
    /* Ignore leading spaces */
    while (*buf && (*buf == ' '))
        buf++;

    /* Build the argv list */
    argc = 0;
    if (*buf == '\'') {
        buf++;
        delim = strchr(buf, '\'');
    }
    else {
        delim = strchr(buf, ' ');
    }

    while(delim) {
        argv[argc++] = buf;
        *delim = '\0';
        buf = delim + 1;
        while (*buf && (*buf == ' '))
            buf++;

        if (*buf == '\'') {
            buf++;
            delim = strchr(buf, '\'');
```

```
        }
        else {
            delim = strchr(buf, ' ');
        }
    }
    argv[argc] = NULL;
    return argc;
}
```

通用头文件

zxcpyp_sys.h – 系统编程头文件

```
/*
  * zxcpyp's lib
  *
  * zxcpyp sys header
  */

#ifndef ZXCPYP_H
#define ZXCPYP_H

#include <stdio.h>          /* Standard I/O functions */
#include <stdlib.h>         /* Standard library */
#include <string.h>         /* String handling functions */
#include <unistd.h>         /* Unix system calls */
#include <errno.h>          /* Error difinations */
#include <fcntl.h>          /* Micros for I/O */
#include <sys/types.h>    /* System data type */
#include <sys/stat.h>     /* File system status */

#include "zxcpyp_err.h" /* Error handling functions */

/* True and false defines */
#ifndef FALSE
    #define FALSE 0
#endif // !FALSE
#ifndef TRUE
    #define TRUE 1
#endif // !TRUE

/* Max and min defines */
#ifndef min
    #define min(m, n) ((m) < (n) ? (m) : (n))
#endif // !min
#ifndef max
    #define max(m, n) ((m) > (n) ? (m) : (n))
#endif // !max

#endif // !ZXCPYP_H
```

zxcpyp_err.h – 错误处理头文件

```
/*
  * zxcpyp's lib
  *
  * zxcpyp error message lib
  */

#include "zxcpyp_err.h"

void err_msg(const char *err) {
    printf("Error: %s\n", err);
}

void err_exit(const char *err) {
    printf("%s error!\n", err);
    printf("Exit with 1.\n");
    exit(1);
}

void usage_err(const char *err) {
    printf("Usage: %s\n", err);
    exit(1);
}

void fatal_err(const char *err) {
    printf("Fatal: %s\n", err);
    exit(1);
}
```

zxcpyp_err.c – 错误处理函数

```
/*
  * zxcpyp's lib
  *
  * zxcpyp error message lib
  */

#include "zxcpyp_err.h"

void err_msg(const char *err) {
    printf("Error: %s\n", err);
}

void err_exit(const char *err) {
    printf("%s error!\n", err);
    printf("Exit with 1.\n");
    exit(1);
}

void usage_err(const char *err) {
    printf("Usage: %s\n", err);
    exit(1);
}
```

```
void fatal_err(const char *err) {
    printf("Fatal: %s\n", err);
    exit(1);
}
```

编译说明

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.9)

project(HUST_OS_design)

# Add GTK2
find_package(PkgConfig REQUIRED)
pkg_check_modules (GTK2 REQUIRED gtk+-2.0)
set(CMAKE_C_STANDARD 11)
include_directories (${GTK2_INCLUDE_DIRS})
link_directories (${GTK2_LIBRARY_DIRS})
add_definitions (${GTK2_CFLAGS_OTHER})

# Add zxcpyp lib
add_subdirectory(./lib)

# Build Part I
add_executable(copy ./PartI-Linux_basics/copy.c)
target_link_libraries(copy lib)
add_executable(fork_demo ./PartI-Linux_basics/fork_demo.c)
target_link_libraries(fork_demo lib ${GTK2_LIBRARIES})

# Build Part II
add_executable(testcall ./PartII-Syscall/testcall.c)
add_executable(teststr ./PartII-Syscall/teststr.c)
add_executable(testcp ./PartII-Syscall/testcp.c)

# Build Part III
add_executable(testdev ./PartIII-Device_driver/testdev.c)

# Build Part IV
aux_source_directory(./PartIV-System_monitor sysmonitor)
add_executable(py_sysmonitor ${sysmonitor})
target_link_libraries(py_sysmonitor lib ${GTK2_LIBRARIES})

# Build Part V

aux_source_directory(./PartV-File_system_simulation fs)
add_executable(zxcpypfs ${fs})
target_link_libraries(py_sysmonitor lib)
```

参考文献

1 庞丽萍，阳富民著. 计算机操作系统（第 2 版）. 北京：人民邮电出版社，2014.

2 [美] Andrew S.Tanenbaum, Herbert Bos 著，陈向群，马洪兵译. 现代操作系统（原书第 4 版）. 北京：机械工业出版社，2017.

3 [德] Michael Kerrisk 著，孙剑，徐从年等译. Linux/UNIX 系统编程手册. 北京：人民邮电出版社，2014.

4 [美] Robert Love 著，陈莉君，康华译. Linux 内核设计与实现（原书第 3 版）. 北京：机械工业出版社，2011.

5 [美] Randal E.Bryant，David O'Hallaron 著，龚奕利，贺莲译. 深入理解计算机系统（原书第 3 版）. 北京：机械工业出版社，2016.

6 Arch Linux 官方 wiki 文档. URL：https://wiki.archlinux.org/

7 Linux 之 GTK 系列教程. 作者：lianghe_work. URL：https://blog.csdn.net/lianghe_work/article/details/47041153