
GENIUX

项目设计与功能说明文档

小组成员：张文喆 155

宋天佑 155

韩乐桐 1552761

班 级：操作系统课程设计（2）班

指导教师：王冬青

完成时间：2018-09-09

目录

一、项目描述	1
1. 项目目的	1
2. 开发环境	1
3. 项目指标	1
二、设计方案	1
1. 概述	1
2. 系统功能结构及使用说明	1
2.1 开机动画效果	1
2.2 帮助页面	2
2.3 进程管理功能	3
2.4 清屏功能	4
2.5 文件管理功能	4
2.5.1 文件管理	4
2.5.2 显示目录文件列表	4
2.5.3 创建文件	5
2.5.4 删除文件	6
2.5.5 打印文件内容	6
2.5.6 写文件	7
2.5.7 创建目录	8
2.5.8 进入多级目录	8
2.6 游戏功能	9
2.6.1 九宫棋小游戏	9
2.6.2 2048小游戏	10
三、项目核心代码	11
1. ProcessManage()	11
2. ls()	11

3. GoDir()	12
4. TTT()	13
5.start2048Game()	13
四、成员分工	15

一、项目描述

1. 项目目的

操作系统是软件专业的一门必修课，但由于操作系统比较抽象，所以一学期结束后还是还很难理解进程、内存、文件管理等一些概念和原理。操作系统课程设计的目的就是通过自己学习、实现一个简单而功能完善的操作系统，来让我们真正理解一个操作系统是如何从无到有、一步步实现的。

2. 开发环境

- a. 基于32位Ubuntu开源GNU/Linux操作系统
- b. 使用Bochs开源模拟器

3. 项目指标

本门操作系统课程设计中，3人为一组，利用《Orange's 一个操作系统的实现》书中提供的源代码，通过修改或者重新实现参考源码的一个或多个模块来实现一个简单的操作系统。

本系统主要针对文件系统进行重新实现，其中新增代码量达到文件模块代码的一半，实现**B级**项目难度；在参考源码的基础上实现系统级应用，如磁盘，工具台等，通过调用较多的系统API以实现系统的检测和控制，实现**C级**项目难度；同时实现了用户级应用，实现**D级**项目难度。

二、设计方案

1. 概述

本系统参考《Orange's 一个操作系统的实现》书中提供的源代码，重新实现了文件系统部分，可以针对文件/目录进行增删改写操作，同时实现了多级目录，可在不同层级目录中进行转换；同时通过调用部分系统API实现九宫棋、2048等系统小游戏，并且增加了特别的开机动画。

2. 系统功能结构及使用说明

2.1 开机动画效果

◆ 描述

进入系统后，代表本操作系统名称的“GENIUX”字样自右至左进入，随后闪出项目组成员姓名及卡通图案，3秒后自动消失。

◆ 实现方法

开机动画使用逐帧printf的方式实现，通过设置较小的延迟时间形成动画效果。



2.2 帮助页面

◆ 描述

开机动画结束后，系统自动显示欢迎界面，用户输入“help”指令后，显示当前指令列表，方便用户进行相应操作；指令输入区“[root@localhost: /]”显示当前路径，“/”表示当前所在目录为根目录。

◆ 实现方法

用户输入指令后，系统通过判断字符串调用对应API，进入相应功能（详见下文）。

```

Bochs x86 emulator, http://bochs.sourceforge.net/

=====
Geniux v1.0.0
Kernel in Orange's

Welcome !

root@localhost: ~# help
=====Geniux help Info=====
Command List
1. process      : A process manager, show you all processes-info here
2. filemg       : Run the file manager
3. clear        : Clear the screen
4. help         : Show this help message
5. ls           : List all files in current directory
6. touch [filename] : Create a new file in current directory
7. rm [filename]  : Delete a file in current directory
8. cat [filename] : Print the content of a file in current directory
9. vi [filename]  : Write new content at the end of the file
10. mkdir [dirname] : Create a new directory in current directory
11. cd [dirname]  : Go to a directory in current directory
12. runttt       : Run a small game on this OS
13. run2048      : Run 2048 game on this OS
=====
root@localhost: ~#
TPS: 89,622H  YES

```

2.3 进程管理功能

◆ 描述

用户输入“process”指令，进入进程管理功能，系统打印出当前全部进程（包括系统进程和用户进程）的pid，名称，优先级（15表示系统级进程，5表示用户级进程）以及运行状态（YES表示正在运行，NO表示没有运行）。

◆ 实现方法

通过调用ProcessManage() API，遍历整个进程列表，逐个打印，同时根据其p_flags是否为FREE_SLOT来判断进程运行状态。

```

root@localhost: ~# process
=====
PID      name      priority  running?
-----
0        TTY       15        YES
1        SYS       15        YES
2        HD        15        YES
3        FS        15        YES
4        MM        15        YES
5        TestA     5         YES
6        TestB     5         YES
7        TestC     5         YES
8                0         NO
9                0         NO
10               0         NO
11               0         NO
12               0         NO
=====

```

2.4 清屏功能

◆ 描述

用户输入“clear”命令后，系统进行清屏，显示上图初始欢迎界面。

◆ 实现方法

通过调用clear() API实现清屏功能，同时重新设置控制台相应初始值。



2.5 文件管理功能

2.5.1 文件管理

◆ 描述

用户输入“filemng”指令，系统告知用户，当前正处于1号控制台，文件管理系统正在运行中。



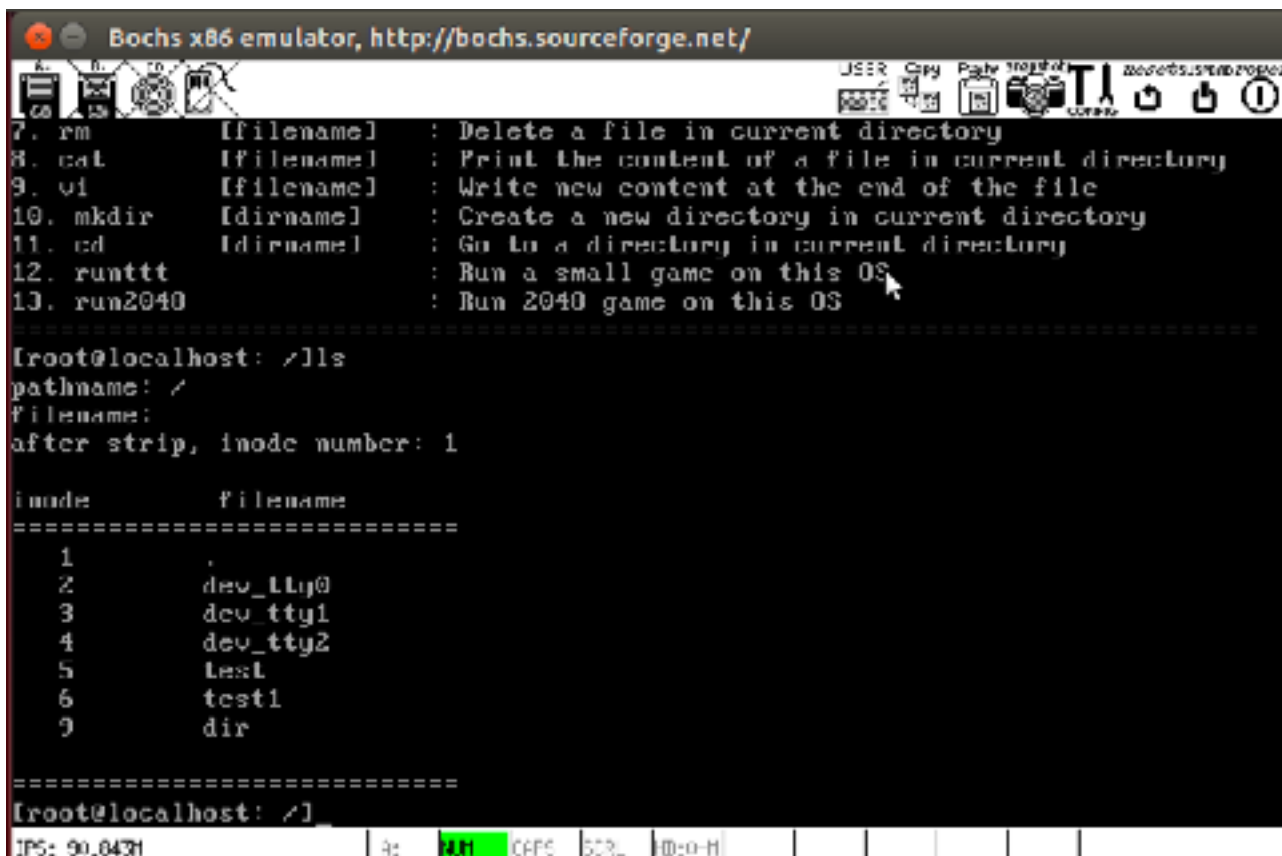
2.5.2 显示目录文件列表

◆ 描述

用户输入“ls”指令，系统打印出当前文件目录下的全部文件及目录。

◆ 实现方法

调用ls()方法，找到当前目录下的全部文件名。



```
Bochs x86 emulator, http://bochs.sourceforge.net/
=====
7. rm      [filename]      : Delete a file in current directory
8. cat     [filename]      : Print the content of a file in current directory
9. vi      [filename]      : Write new content at the end of the file
10. mkdir  [dirname]       : Create a new directory in current directory
11. cd     [dirname]       : Go to a directory in current directory
12. runttt                : Run a small game on this OS
13. run2040               : Run 2040 game on this OS
=====
[root@localhost: ~]ls
pathname: /
filename:
after strip, inode number: 1

inode      filename
=====
1          .
2          dev_tty0
3          dev_tty1
4          dev_tty2
5          test1
6          test1
7          dir
=====
[root@localhost: ~]
```

2.5.3 创建文件

◆ 描述

用户输入“touch + filename”指令，在当前目录下创建一个新的文件并命名，若创建成功则系统提示“file created”的消息，若创建失败则系统提示“failed to create a new file”的消息（如上图）。

◆ 实现方法

通过调用open()方法创建新文件，根据方法返回值判断是否创建成功并给用户发出反馈消息。

```
=====
[root@localhost: ~]touch myfile
pathname: /myfile
filename:
pathname: /myfile
filename:
File created: myfile (fd 2)
[root@localhost: ~]
```



```

Bochs x86 emulator, http://bochs.sourceforge.net/
File created: myfile (fd 2)
[root@localhost: /]ls
pathname: /
filename:
after strip, inode number: 1

inode      filename
=====
 1         .
 2         dev_tty0
 3         dev_tty1
 4         dev_tty2
 5         test
 6         test1
 9         dlr
10         mydir
11         myfile

=====
[root@localhost: /]touch myfile
pathname: /myfile
filename:
file exists.
Failed to create a new file with name myfile
[root@localhost: /]

```

2.5.4 删除文件

◆ 描述

用户输入“rm+filename”指令，删除指定文件名的文件（由于不允许相同目录下文件重名，所以此处可以使用文件名进行搜索），删除成功后系统提示“deleted”，若删除失败（当前目录下不存在该文件）则提示“failed to delete file”。

◆ 实现方法

调用unlink() API实现文件删除。

```

[root@localhost: /]rm myfile
pathname: /myfile
filename:
pathname: /myfile
filename: G^@
myfile deleted!
[root@localhost: /]

```

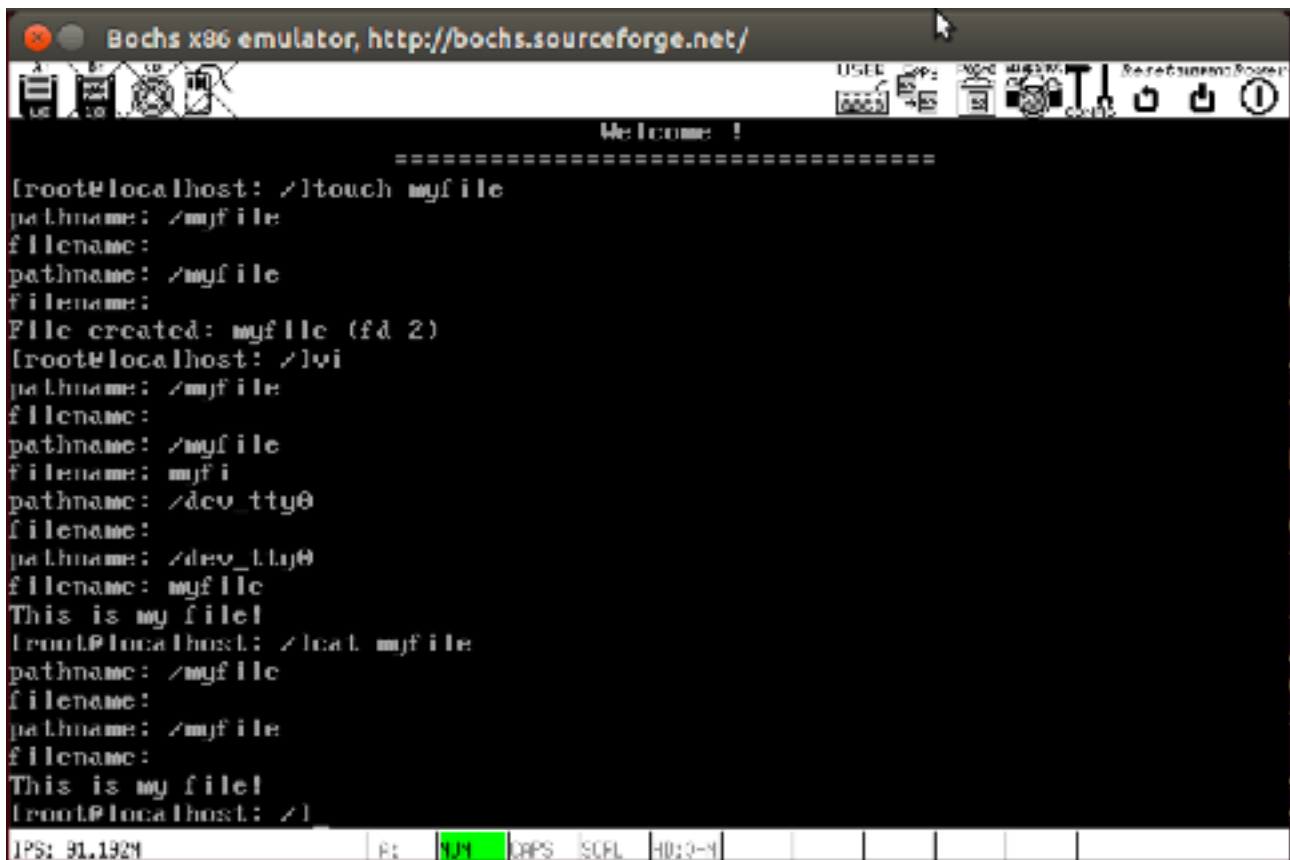
2.5.5 打印文件内容

◆ 描述

用户输入“cat+filename”指令，查看该文件内容，若文件中已写入内容，则如上图显示；若文件创建后未写入内容，则打印空行；若找不到文件则显示“fail to open”；若找到文件后打开失败则显示“An error has occurred in reading the file”。

◆ 实现方法

首先使用open()方法定位文件，然后调用read()方法对文件进行读操作。



```
Bochs x86 emulator, http://bochs.sourceforge.net/

Welcome !

=====
[root@localhost: ~]# touch myfile
pathname: /myfile
filename:
pathname: /myfile
filename:
File created: myfile (fd 2)
[root@localhost: ~]# vi
pathname: /myfile
filename:
pathname: myfile
pathname: /dev/tty0
filename:
pathname: /dev/tty0
filename: myfile
This is my file!
[root@localhost: ~]# cat myfile
pathname: /myfile
filename:
pathname: /myfile
filename:
This is my file!
[root@localhost: ~]#
```

2.5.6 写文件

◆ 描述

用户输入“vi+filename”指令，系统在定位到目标文件后显示文件路径，之后用户输入要写入文件的内容并回车，完成写文件操作。

◆ 实现方法

首先使用open()方法定位文件，然后调用write()方法对文件进行写操作。

```
[root@localhost: ~]# vi
pathname: /myfile
filename:
pathname: /myfile
filename: myfile
pathname: /dev/tty0
filename:
pathname: /dev/tty0
filename: myfile
This is my file!
[root@localhost: ~]#
```

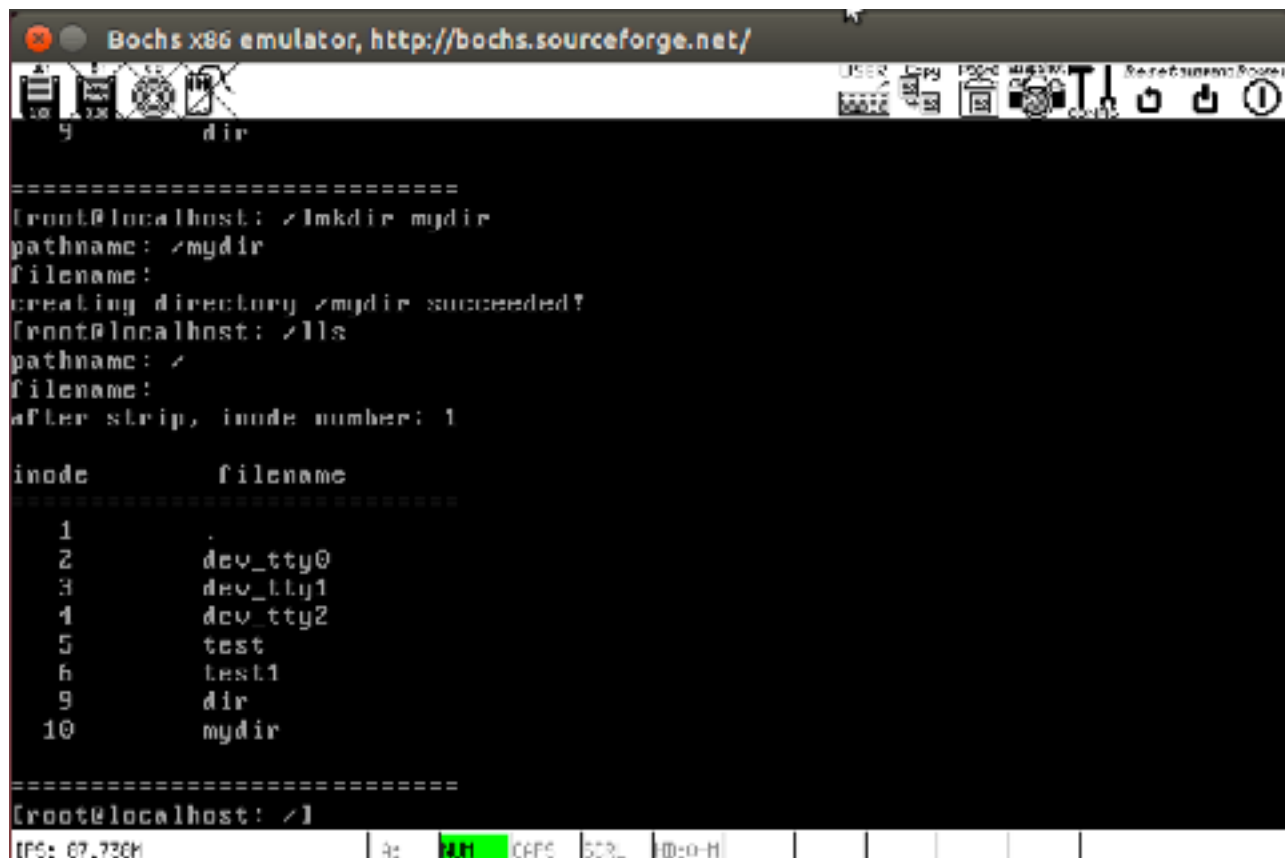
2.5.7 创建目录

◆ 描述

用户输入“mkdir + directoryname”指令，在当前目录内创建一个目录。

◆ 实现方法

首先使用open()方法创建对应文件路径，然后调用mkdir()方法，创建目录，成功或失败都会有相应提示信息。



```
Bochs x86 emulator, http://bochs.sourceforge.net/
=====
[ root@localhost: ~ ] mkdir mydir
pathname: ./mydir
filename:
creating directory ./mydir succeeded!
[ root@localhost: ~ ] ll
pathname: ./
filename:
after strip, inode number: 1

inode      filename
=====
1          .
2          dev_tty0
3          dev_tty1
4          dev_tty2
5          test
6          test1
9          dir
10         mydir

=====
[ root@localhost: ~ ]
```

2.5.8 进入多级目录

◆ 描述

用户输入“cd + directoryname”指令，系统自动检索目录路径，检索成功后，指令输入区的路径由“/”更改为“/directoryname/”，表示当前处于目标目录中。

◆ 实现方法

调用GoDir()方法，首先根据输入的目录名判断要进入下一级目录还是返回上一级目录，若返回上一级，则先找到目标位置，再将绝对路径修改为对应路径并更新；若进入下一级，则直接修改路径并更新。

```

Bochs x86 emulator, http://bochs.sourceforge.net/
creating directory /mydir: succeeded!
[root@localhost: /]ls
pathname: /
filename:
after strip, inode number: 1

inode      filename
=====
1          .
2          dev_tty0
3          dev_tty1
4          dev_tty2
5          test
6          test1
7          dir
10         mydir

=====
[root@localhost: /]cd mydir
test::
pathname: /mydir/
filename:
pathname: /mydir/
filename:
[root@localhost: /mydir/]
IPS: 89,4454  CPU: 100%

```

2.6 游戏功能

2.6.1 九宫棋小游戏

◆ 描述

用户输入“runttt”指令，进入九宫棋游戏界面；首先系统打印空白棋盘（QP），然后用户通过输入棋子行、列位置来与系统下棋，直至出现死局或赢家；用户中途可以输入回车直接退出游戏界面，返回初始欢迎界面。

◆ 实现方法

调用TTT()方法，其中调用多个九宫棋相关函数（如UserInput(), AutoDone()等），实现九宫棋游戏的相应操作。

```

Bochs x86 emulator, http://bochs.sourceforge.net/
=====
[root@localhost: ~]# ./runqip
The QIPan (QP) is:
  0  0  0
  0  0  0
  0  0  0
Do you want to firstly?(y/n):n
The computer put a Chessman at: 2,2
The QP now is:
  0  0  0
  0  0  0
  0  0  0
Please Input The Line Position where you put your Chessman (x): 1
Please Input The Column Position where you put your Chessman (y): 2
The computer put a Chessman at: 3,3
The QP now is:
  0  0  0
  0  0  0
  0  0  0
Please Input The Line Position where you put your Chessman (x): 1
Please Input The Column Position where you put your Chessman (y): 1
The computer put a Chessman at: 1,3
The QP now is:
  0  0  0
  0  0  0
  0  0  0
=====
IPS: 93.551M  F: 104  OPS  SCPL  HD:2-M

```

2.6.2 2048小游戏

◆ 描述

```

Welcome to 2048 Game!!!

Controls:
LEFT: a    RIGHT: d
UP:   w    DOWN: s
EXIT: press enter

-----
| | | | |
| | | | |
| | | | |
| 2 | | | |
|-----|
| 4 | 8 | 16 |
|-----|
Score: 60

```

用户输入“run2048”指令，进入2048游戏界面；系统自动打印游戏控制方法（wsad控制上下左右，enter键退出游戏）以及初始棋盘、当前分数（初始为0）；用户每次只能输入一个运动方向，系统自动刷新棋盘及分数，同时在棋盘内随机生成新数字格。

◆ 实现方法

调用start2048Game()方法，其中调用多个2048相关函数（如morge2048(), printNums2048, addRandom2048()等），共同实现2048游戏的操作。

三、项目核心代码

1. ProcessManage()

```
void ProcessManage()
{
    int i;

    printf("=====\n");
    printf("      PID      |   name      | spriority   | running?\n");
    //进程号, 进程名, 优先级, 是否是系统进程, 是否在运行

    printf("-----\n");
    for ( i = 0 ; i < NR_TASKS + NR_PROCS ; ++i )//逐个遍历
    {
        /*if ( proc_table[i].priority == 0) continue;//系统资源跳过*/
        printf("      %d      %s      %d      %s\n",
proc_table[i].pid, proc_table[i].name, proc_table[i].priority,
proc_table[i].p_flags==FREE_SLOT? "NO":"YES");
    }

    printf("=====\n");
}
```

2. ls()

```
PUBLIC int ls(char* pathName) // 传入当前目录, 发送当前目录下的文件名
{
    MESSAGE msg;
    msg.type = LS; // ls类型的消息

    msg.PATHNAME = (void*)pathName;
    msg.NAME_LEN = strlen(pathName);
    msg.FLAGS = 0;

    send_recv(BOTH, TASK_FS, &msg);

    return msg.RETVAL;
}
```

3. GoDir()

```

void GoDir(char* path, char* file)
{
    int flag = 0; // 判断是进入下一级目录还是返回上一级目录
    char newPath[512] = {0};
    if (file[0] == '.' && file[1] == '.') // cd ..返回上一级目录
    {
        flag = 1;
        int pos_path = 0;
        int pos_new = 0;
        int i = 0;
        char temp[128] = {0}; // 用于存放某一级目录的名称
        while (path[pos_path] != 0)
        {
            if (path[pos_path] == '/')
            {
                pos_path++;
                if (path[pos_path] == 0) // 已到达结尾
                    break;
                else
                {
                    temp[i] = '/';
                    temp[i + 1] = 0;
                    i = 0;
                    while (temp[i] != 0)
                    {
                        newPath[pos_new] = temp[i];
                        temp[i] = 0; // 抹掉
                        pos_new++;
                        i++;
                    }
                    i = 0;
                }
            }
            else
            {
                temp[i] = path[pos_path];
                i++;
                pos_path++;
            }
        }
    }
    char absoPath[512];
    char temp[512];
    int pos = 0;
    printf("test::\n");
    while (file[pos] != 0)
    {
        temp[pos] = file[pos];
        pos++;
    }
    temp[pos] = '/';
    temp[pos + 1] = 0;
    if (flag == 1) // 返回上一级目录
    {
        temp[0] = 0;
        convert_to_absolute(absoPath, newPath, temp);
    }
    else // 进入下一级目录
        convert_to_absolute(absoPath, path, temp);
    int fd = open(absoPath, O_RDWR);
    if (fd == -1)
        printf("%s is not a directory!\n", absoPath);
    else
        memcpy(path, absoPath, 512);
}

```

4. TTT()

```

void TTT(int fd_stdin,int fd_stdout)
{
    char buf[80]={0};
    char IsFirst = 0;
    int IsFinish = FALSE;
    while(!IsFinish)
    {

        Init();
        printf("The QiPan (QP) is: \n");

        PrintQP();

        printf("Do you want do first?(y/n):");
        read(fd_stdin,buf,2);
        IsFirst = buf[0];
        do{

            if(IsFirst=='y')
            {
                UserInput(fd_stdin, fd_stdout);
                IsFinish=AutoDone();
            }else{
                IsFinish=AutoDone();
                if(!IsFinish)UserInput(fd_stdin, fd_stdout);
            }

        }while(!IsFinish);
        if (IsFinish)
        {
            printf("Play Again?(y/n)");
            char cResult;
            read(fd_stdin,buf,2);
            cResult = buf[0];
            printf("%c",cResult);
            if (cResult == 'y')
            {
                clear();
                IsFinish = FALSE;
            }
            else
            {
                clear();
            }
        }
    }
}

```

5.start2048Game()

```

PUBLIC void start2048Game(int fd_stdin, int fd_stdout) {
    // Specify the rules of the game

    clear();
    printf("Welcome to 2048 Game!\n\n\n");
    printf("Control:\n");
    printf("          LEFT: a    RIGHT: d\n");
    printf("          UP:   w    DOWN: s\n");
    printf("          EXIT: press enter \n\n\n");
    // Initialize the data
    initData();
}

```



```

// Initalization
addrandom2048();
addrandom2048();
printNums2048();

// Turns in loops
// while (scanf(" %c", &option)) {
while (read(fd_stdin, option2048, 2)) {

    clear();
    printf("Welcome to 2048 Game!!!\n\n\n");
    printf("Controls:\n");
    printf("          LEFT: a    RIGHT: d\n");
    printf("          UP:   w    DOWN: s\n");
    printf("          EXIT: press enter  \n\n\n");
    // Check if the player is dead
    if (!isAlive2048()) {
        printf("You lose!!!\a\n");
        break;
    }

    morge2048();
    if (validity2048) {
        addrandom2048();
    }
    validity2048 = 0;

    printNums2048();
}
clear();
}

void morge2048(void) {
    /* Morges(moves and merges) the number blocks */

    switch (option2048[0]) {
    case 'w':
        for (int j = 0; j <= 3; j++) {
            for (int i = 0, k = 0; i <= 3; i++) {
                tempUnit2048[k++] = numbers2048[i][j];
            }

            move2048();
            merge2048();
            move2048();

            for (int i = 0, k = 0; i <= 3; i++) {
                numbers2048[i][j] = tempUnit2048[k++];
            }
        }
        break;
    case 'a':
        for (int i = 0; i <= 3; i++) {
            for (int j = 0, k = 0; j <= 3; j++) {
                tempUnit2048[k++] = numbers2048[i][j];
            }

            move2048();
            merge2048();
            move2048();

            for (int j = 0, k = 0; j <= 3; j++) {
                numbers2048[i][j] = tempUnit2048[k++];
            }
        }
        break;
    case 's':
        for (int j = 0; j <= 3; j++) {
            for (int i = 3, k = 0; i >= 0; i--) {
                tempUnit2048[k++] = numbers2048[i][j];
            }
        }
    }
}

```

```
        move2048();
        merge2048();
        move2048();

        for (int i = 3, k = 0; i >= 0; i--) {
            numbers2048[i][j] = tempUnit2048[k++];
        }
        break;
    case 'd':
        for (int i = 0; i <= 3; i++) {
            for (int j = 3, k = 0; j >= 0; j--) {
                tempUnit2048[k++] = numbers2048[i][j];
            }

            move2048();
            merge2048();
            move2048();

            for (int j = 3, k = 0; j >= 0; j--) {
                numbers2048[i][j] = tempUnit2048[k++];
            }
        }
        break;
    default:
        printf("Illegal input!!!\a\n");
    }
    return;
}
```

四、成员分工

姓名	学号	分工	占比
张文喆	1551719	命令行功能实现，多级文件系统实现	33.3%
宋天佑	1551177	开机动画、游戏功能实现，文件系统实现	33.3%
韩乐桐	1552761	设计与功能文档编写，命令行功能实现	33.3%