项目说明文档

操作系统

——OrangeKingOS

组 长： 黄开宇（1952647）

组 员 一： 娄天宇（1950067）

组 员 二： 王 凌（1951504）

指 导 教 师： 王冬青

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 项目概述 1](#_Toc79411893)

[1.1 项目简介 1](#_Toc79411894)

[1.2 项目配置 1](#_Toc79411895)

[1.3 项目分工 1](#_Toc79411896)

[2 操作说明 2](#_Toc79411897)

[2.1 界面初始化动画 2](#_Toc79411898)

[2.2 系统级应用——控制台命令行系统 2](#_Toc79411899)

[2.3 系统级应用——进程管理 2](#_Toc79411900)

[2.4 系统级应用——文件管理 5](#_Toc79411901)

[2.5 用户级应用——日历 11](#_Toc79411902)

[2.6 用户级应用——计算器 17](#_Toc79411903)

[2.7 用户级应用——2048 18](#_Toc79411904)

[3 功能实现 20](#_Toc79411905)

[3.1 界面初始化动画 20](#_Toc79411906)

[3.2 系统级应用——控制台命令行系统 21](#_Toc79411907)

[3.3 系统级应用——进程管理 23](#_Toc79411908)

[3.4 系统级应用——文件管理 27](#_Toc79411909)

[3.5 用户级应用——日历 30](#_Toc79411910)

[3.6 用户级应用——计算器 36](#_Toc79411911)

[3.7 用户级应用——2048 40](#_Toc79411912)

# 1 项目概述

## 1.1 项目简介

团队以 Orange’s操作系统作为基本框架，对源码进行阅读分析后形成本项目。本项目包括3个系统级应用：控制台命令行系统、进程管理和文件管理，3个用户级应用：计算器，日历和2048游戏，每个应用之下都包含一个或多个实用性较强的功能。此外本项目还包括了几个基本系统函数，如界面初始化动画和清屏等。

用户可以通过在控制台命令行系统中输入命令，使系统将根据输入的命令进行不同的操作。

## 1.2 项目配置

编写语言：汇编语言，C语言

开发环境：Linux系统下的Bochs虚拟机（版本2.6.9）

运行环境：Windows系统下的VMWare Workstation Pro（版本16.0）

## 1.3 项目分工

黄开宇：界面初始化动画、控制台命令行系统、日历

娄天宇：框架搭建、进程管理、计算器

王 凌：文件管理、2048

# 2 操作说明

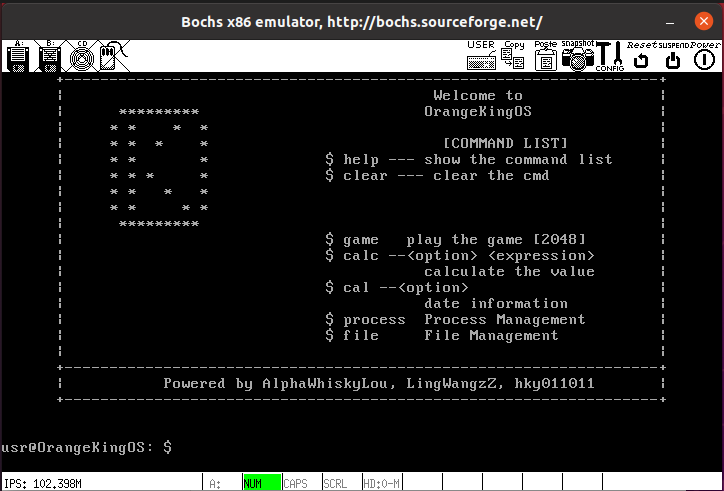
## 2.1 界面初始化动画

界面初始化是逐帧动画，共六帧。之后进入加载界面。加载后接入控制台命令行系统。

## 2.2 系统级应用——控制台命令行系统

系统主界面为home界面，界面显示系统提供的所有功能命令及说明。输入不同命令，即可进入相应功能界面。若想要回到主界面（home界面），在命令行中输入“help”并按下回车即可。一个命令由应用参数（如cal）、功能参数（如—year，以—开头）和附加参数构成。

home界面



## 2.3 系统级应用——进程管理

进程管理子系统的主角面为Process-Management界面，界面显示了进程管理子系统提供的所有功能命令及说明。按照命令提示输入相应命令即可执行相应操作。

ps：展示系统当前所有进程信息。

kill <id>：结束目标进程。若目标进程不存在或是无法被终止则会弹出提示信息。

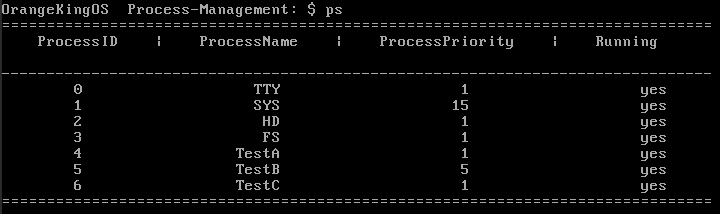
restart <id>：启动未运行进程。若目标进程不存在或是无法被终止则会弹出提示信息。

基本界面：



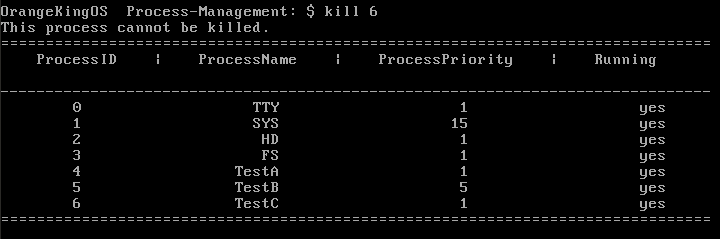
展示当前所有进程状态:

ps



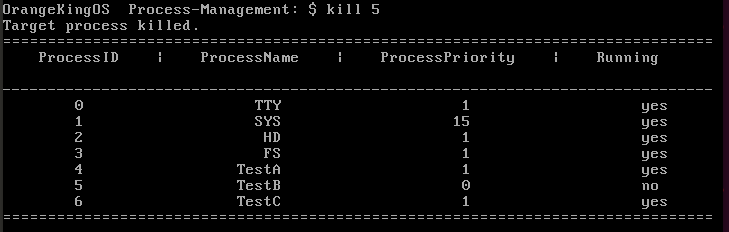
结束指定进程（不能被结束的进程）：

kill 6



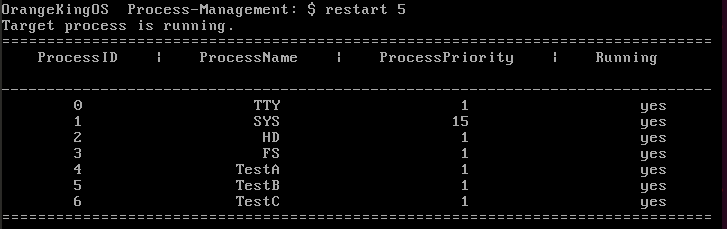
结束指定进程（可以被结束的进程）：

kill 5

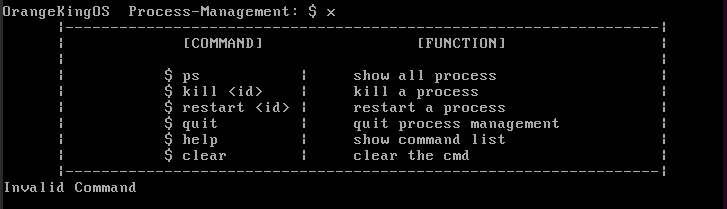


重启指定进程

restart 5



无效指令提示：



## 2.4 系统级应用——文件管理

文件管理子系统的主界面为File-Manager界面，界面显示了文件管理子系统提供的所有功能命令及说明。按照命令提示输入相应命令即可执行相应操作。该指令依照linux系统仿写，但是创建文件由touch改为mkf，且删减了一些指令。

mkf <filename>：make file创建文件。

mkd <dirname>：make directory创建目录。

ls：list的简写，列出该级文件目录下所有文件。

cd <dirname>：change directory的简写，改变目录到dirname。

cd ..：返回上一级目录，在linux系统中，“.”是当前目录，“..”是上一级目录。

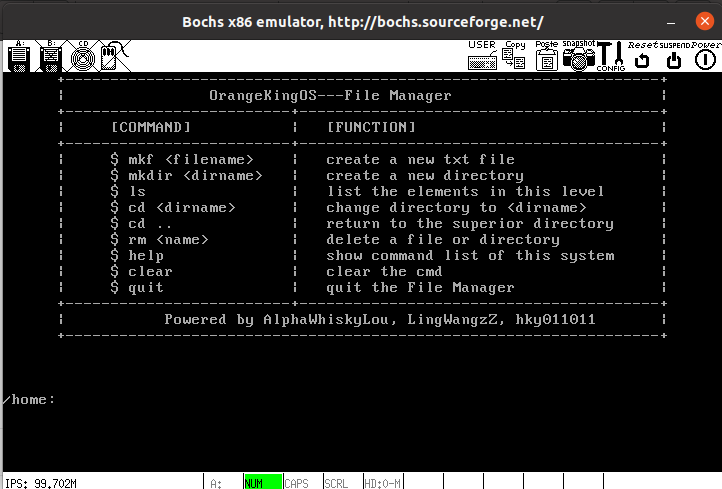
rm <name>：remove的缩写，删除名叫name的文件或者目录

help：帮助，刷新展示File-Manager的主界面

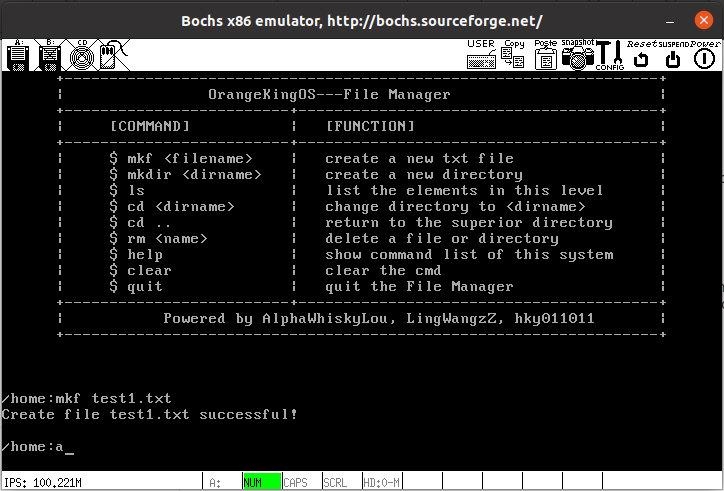
clear：清空command指令

quit：退出

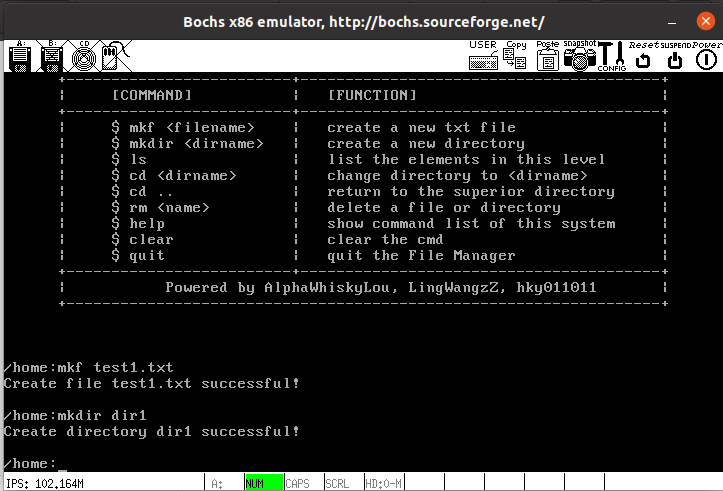
基本界面：



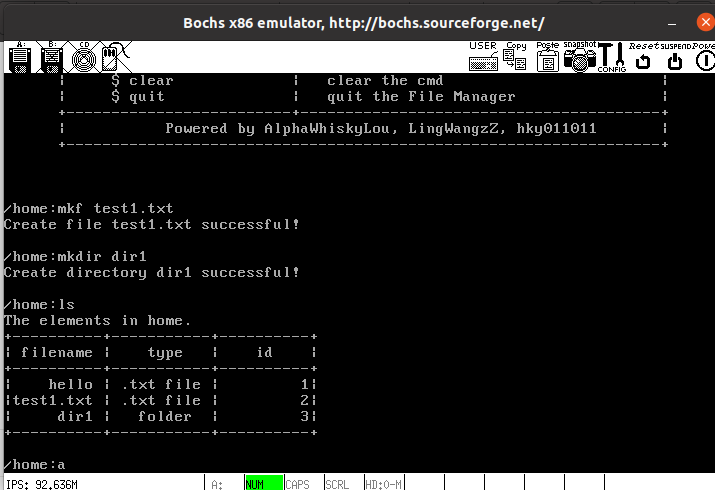
创建文件：



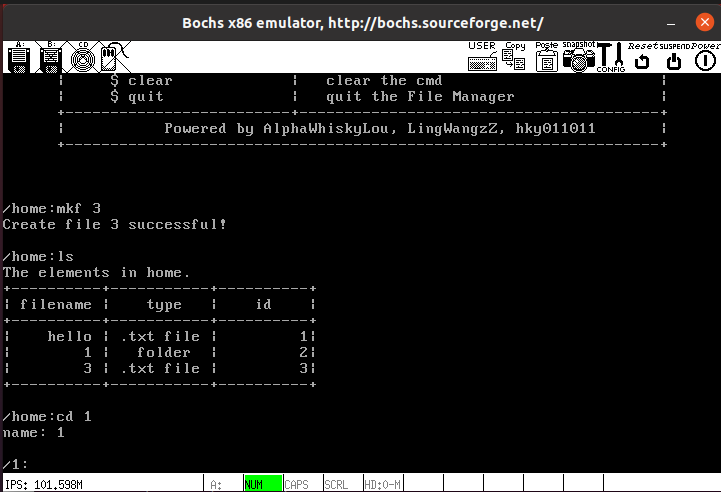
创建目录：



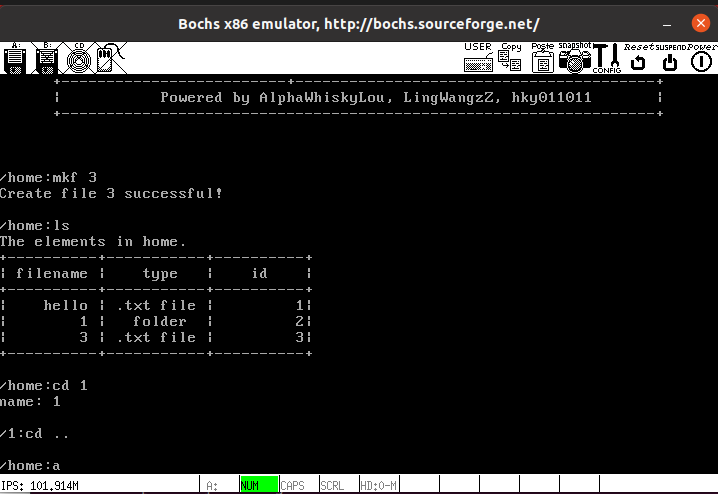
查看当前目录下所有文件：



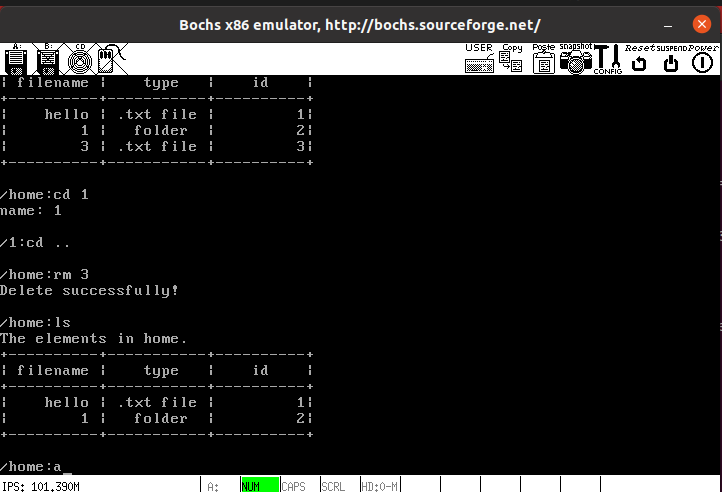
打开其他目录：



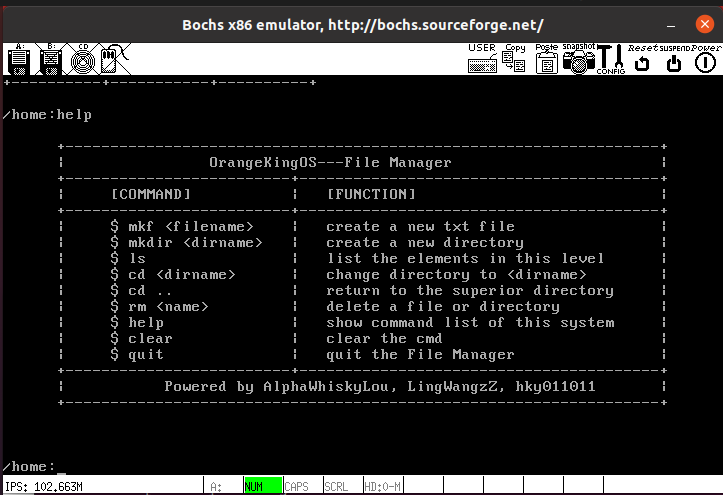
返回上级目录：



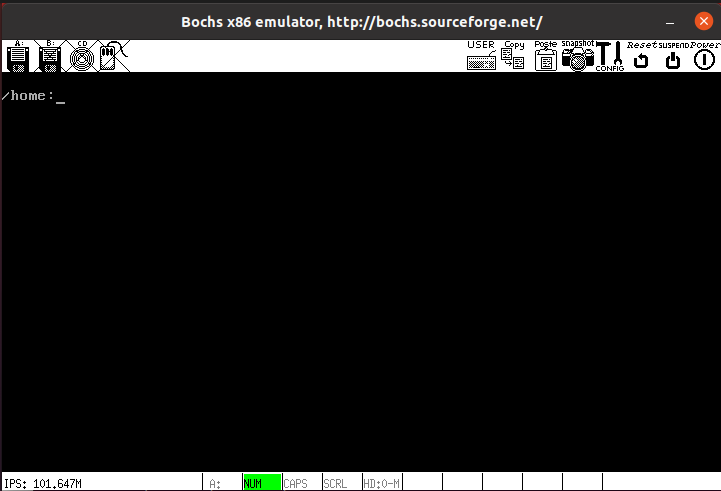
删除文件或者目录：



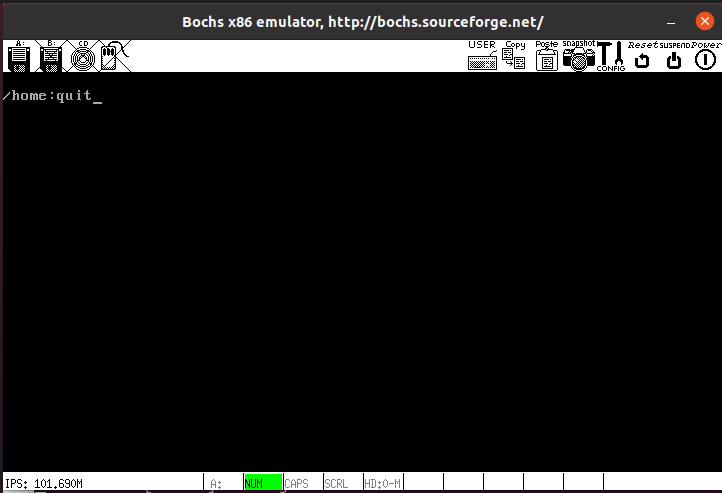
help指令：



clear指令：



quit指令：

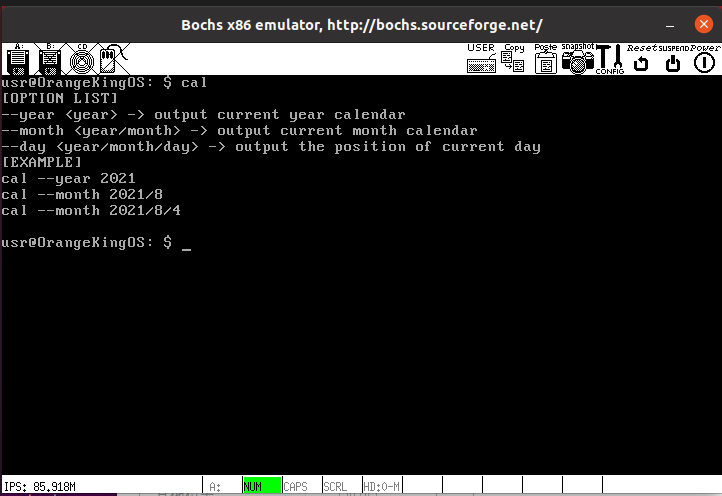




## 2.5 用户级应用——日历

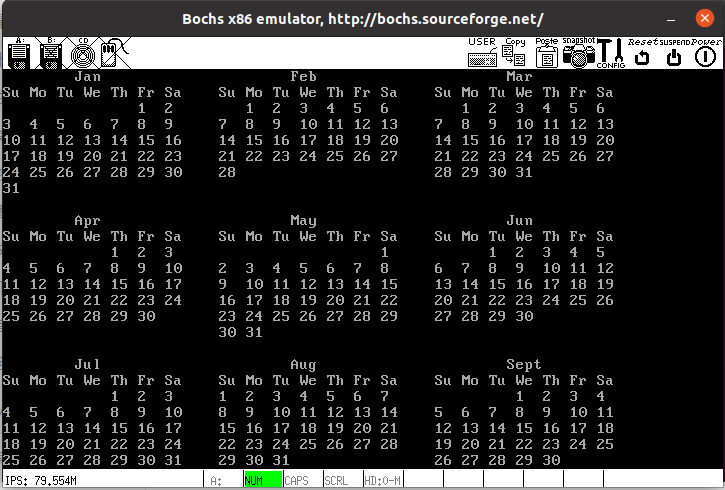
本功能包含三种次级日历功能，能够显示某年年历、某月月历和某日在该年的位置和星期。本功能会对输入进行错误处理，包括年份检测、月份检测、日期检测、闰年检测。

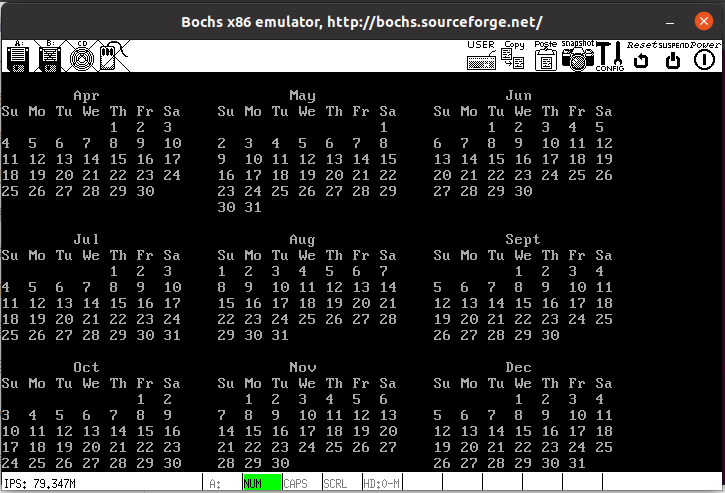
基本功能提示



显示某年年历（可以使用shift + up/down上下翻页）

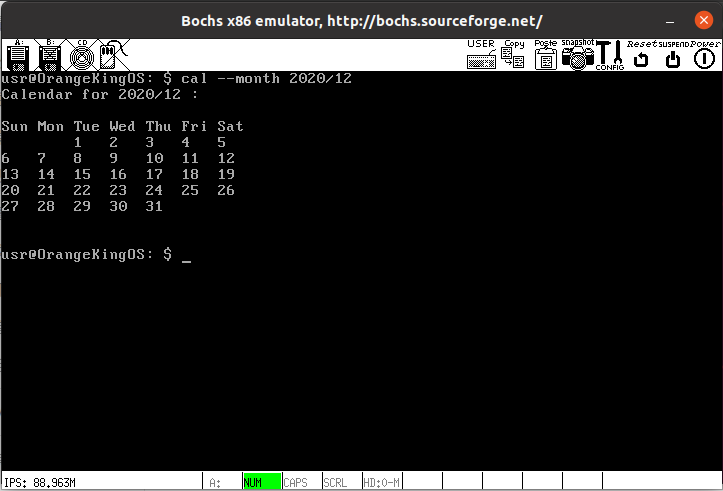
cal --year 2021





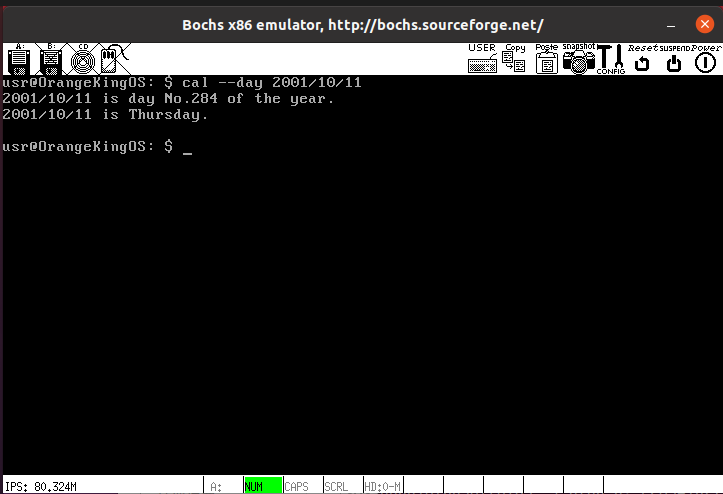
显示某月月历

cal --month 2020/12



显示某日在该年的位置和星期

cal --day 2001/10/11



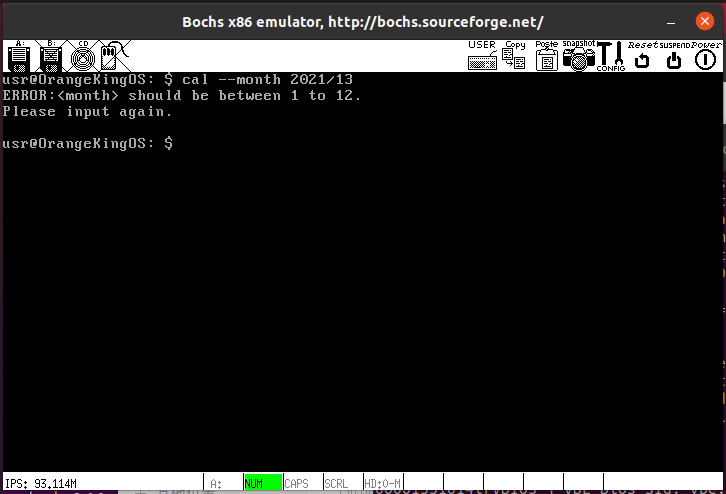
年份检测

cal --year -1



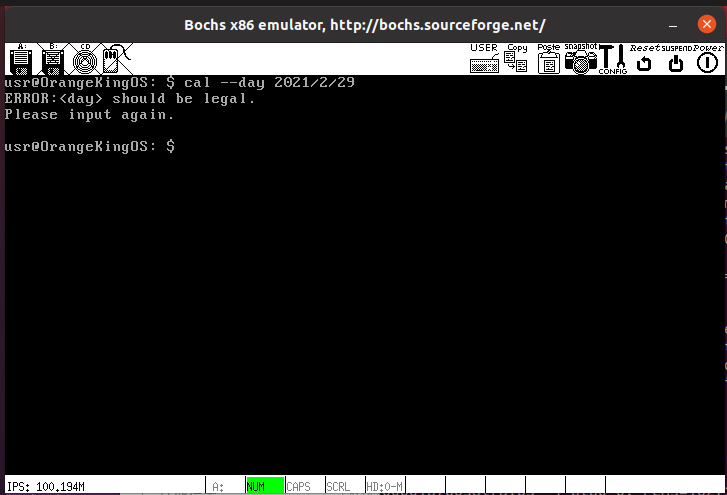
月份检测

cal --month 2021/13



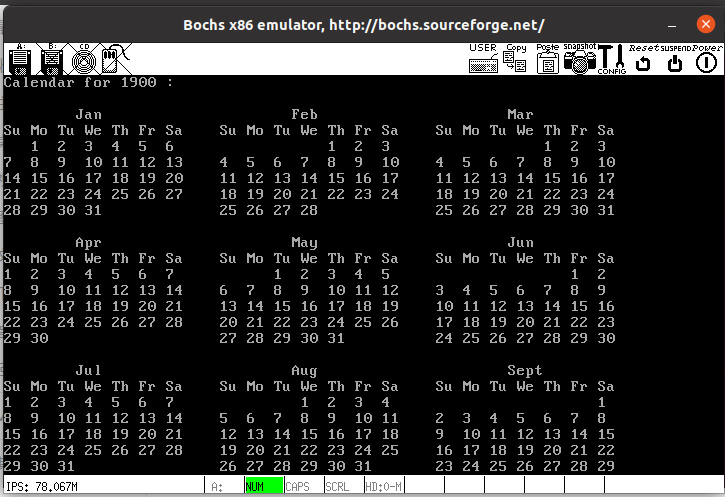
日期检测

cal --day 2021/2/29



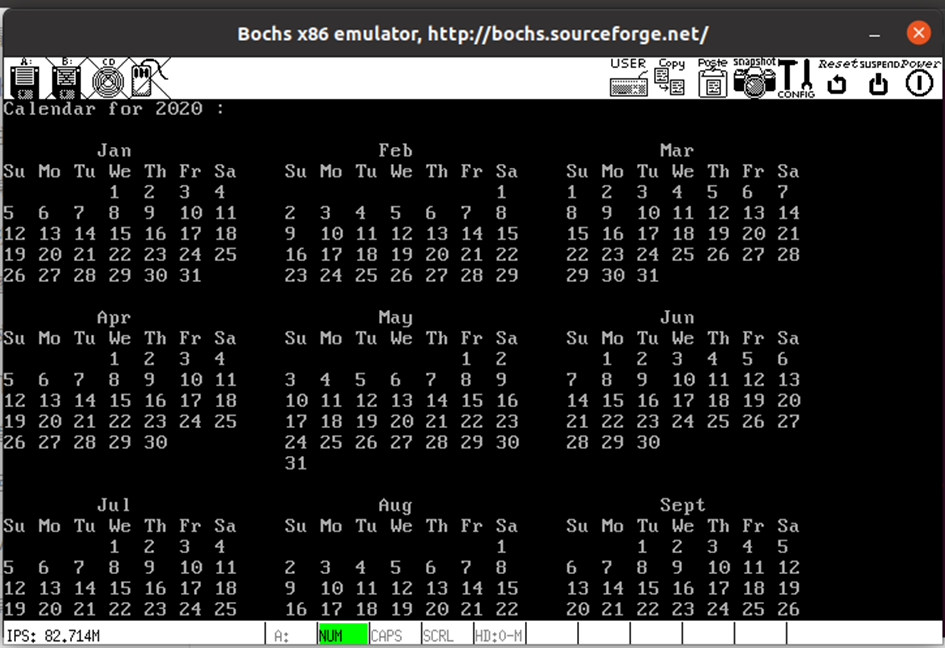
平年二月份检测

cal --year 1900



闰年二月份检测

cal --year 2020

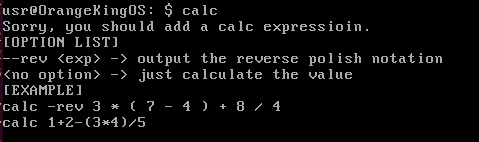


## 2.6 用户级应用——计算器

本功能包含两种计算器次级功能，能够对输入的数学表达式进行计算，并能够向用户输出转换出的后缀表达式。本功能支持括号输入，双目运算符如：加、减、乘、除运算。本功能会对输入发数学表达式进行基本法的错误处理，能够进行表达式中非法字符检测，括号匹配检测。

基本功能提示:

calc



计算表达式的值：

calc 2\*(3+4)



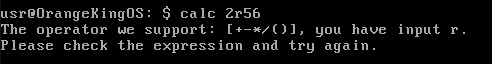
将中缀表达式转换为后缀表达式：

calc --rev 2\*(2+4)



表达式中含有非法字符：

calc 2r56



表达式括号不匹配：

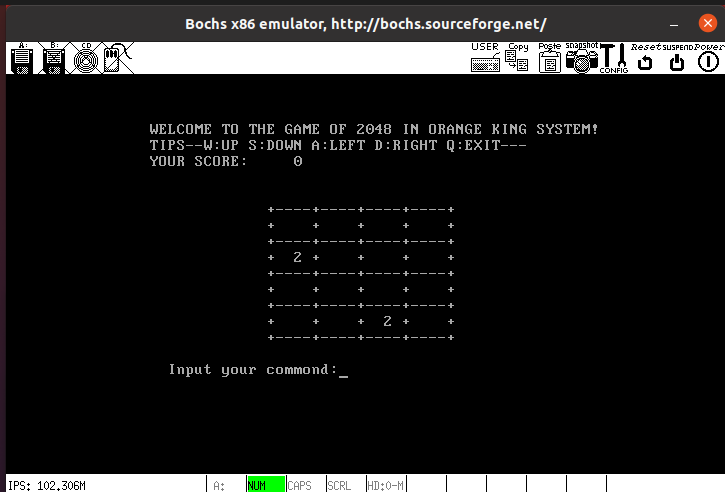
calc 2\*(2+3))



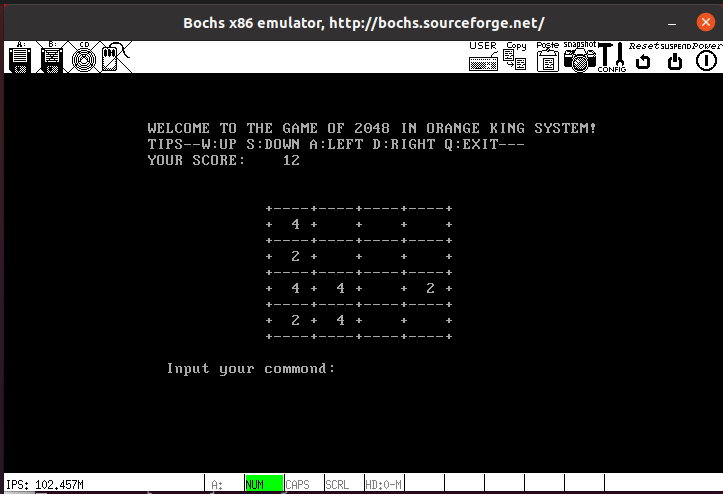
## 2.7 用户级应用——2048

本功能实现了曾经风靡全球的2048游戏，通过 wsad四个按键进行上下左右移动的操作。游戏过程中给出实时分数，并可通过 q 退出。

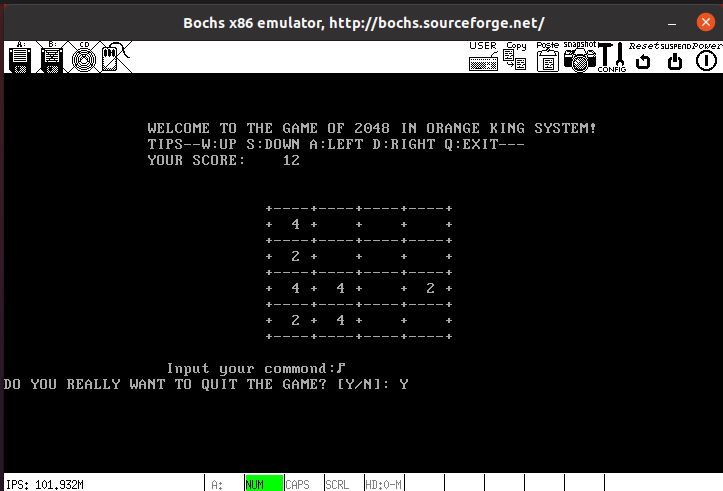
主界面：



游戏过程：



退出：





# 3 功能实现

## 3.1 界面初始化动画

界面初始化动画是逐帧动画，通过反复调用输出该帧页面的函数和clear函数以实现逐帧变换闪烁的样式。

输出该帧页面的函数仅由printf函数组成。

milli\_delay函数

PUBLIC void milli\_delay(int milli\_sec)

{

/\* 记录时间 \*/

int t = get\_ticks();

/\* 循环取时间直到达到延迟时间 \*/

while(((get\_ticks() - t) \* 1000 / HZ) < milli\_sec) {}

}

gradualStart函数（部分）

void gradualStart()

{

printf(" +------------------------------------------------------------------+\n");

printf(" | |\n");

printf(" | |\n");

printf(" | @@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@@@@@@@@@ |\n");

printf(" | @@@@@@@@@ |\n");

printf(" | |\n");

printf(" | |\n");

printf(" +------------------------------------------------------------------+\n");

milli\_delay(DELAY\_TIME);

clear();

## 3.2 系统级应用——控制台命令行系统

为了控制操作系统内核的大小，本项目以read函数替代scanf函数。将键盘输入的字符串作为文件，用read函数读入字符串rdbuf。

此时控制台命令分析转换为分析字符串rdbuf，分析时为做模式匹配必须采用strcmp函数。分析过程为：先采用自己实现的mystrncpy函数取出不同长度的应用参数（如cal取3，math取4），再将剩余功能参数传入各自功能的main函数中进一步分析；对字符串进行扫描，与预设的功能参数（如--year）逐字符比较，并记录是否比较成功。根据比较结果再进入不同功能的代码段，对附加参数（如年份2021）进行错误处理。

控制台命令行跳转

void TestA()

{

int fd;

int i, n;

char tty\_name[] = "/dev\_tty0";

char rdbuf[128];

char command3[100], command4[100], command5[100];

int fd\_stdin = open(tty\_name, O\_RDWR);

assert(fd\_stdin == 0);

int fd\_stdout = open(tty\_name, O\_RDWR);

assert(fd\_stdout == 1);

// char filename[MAX\_FILENAME\_LEN+1] = "zsp01";

const char bufw[80] = { 0 };

clear();

Booting();

Loading();

clear();

CommandList();

while (1) {

printf("usr@OrangeKingOS: $ ");

memset(command3, 0, sizeof(command3));

memset(command4, 0, sizeof(command4));

memset(command5, 0, sizeof(command5));

int r = read(fd\_stdin, rdbuf, 70);

rdbuf[r] = 0;

mystrncpy(command3, rdbuf, 3);

mystrncpy(command4, rdbuf, 4);

mystrncpy(command5, rdbuf, 5);

if (!strcmp(command4, "help"))

{

clear();

CommandList();

}

else if (!strcmp(command5, "clear"))

{

clear();

}

else if (!strcmp(command4, "game"))

{

if (strlen(rdbuf) > 5) {

gameMain(rdbuf + 5, fd\_stdin, fd\_stdout);

}

else {

char\* str = "NULL";

gameMain(str, fd\_stdin, fd\_stdout);

}

continue;

}

else if (!strcmp(command4, "calc"))

{

if (strlen(rdbuf) > 5) {

calcMain(rdbuf + 5);

}

else {

char\* str = "NULL";

calcMain(str);

}

continue;

}

else if (!strcmp(command3, "cal"))

{

if (strlen(rdbuf) > 4) {

calMain(rdbuf + 4);

}

else {

char\* str = "NULL";

calMain(str);

}

continue;

}

else if (!strcmp(rdbuf, "process")) {

clear();

ProcessManagement(fd\_stdin);

}

else if (!strcmp(rdbuf, "file")) {

clear();

runFileManage(fd\_stdin);

}

else if (!strcmp(rdbuf, ""))

{

continue;

}

else

{

clear();

NotFound();

}

}

}

clear函数

/\* 清屏函数 \*/

void clear()

{

/\* 清屏 \*/

clear\_screen(0, console\_table[current\_console].cursor);

/\* 初始化 \*/

console\_table[current\_console].crtc\_start = 0;

console\_table[current\_console].cursor = 0;

}

## 3.3 系统级应用——进程管理

进程管理子系统主要包括了进程当前状态的展示，指定进程的终止与重启。

当前进程状态展示通过读取系统进程表并按照格式输出到屏幕；指定进程的终止通过读取指定进程ID的字符串，得到其ID的数值，并继而以此访问系统进程表将其对应位置的p\_flags修改为-1实现进程终止；指定进程的重启通过读取指定进程ID的字符串，得到其ID的数值，并继而以此访问系统进程表将其对应位置的p\_flags修改为1实现进程重启。

主要函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 返回值类型 | 函数参数 | 作用 |
| ProcessManagement | void | int | 进程管理主函数 |
| showProcessWelcome | void | void | 打印进程管理子系统首页 |
| showProcess | void | void | 展示当前所有进程状态 |
| getPid | int | char[] | 计算进程pid |
| killProcess | void | char[] | 结束指定进程 |
| restartProcess | void | char[] | 重启指定进程 |

核心代码：

//打印所有进程

void showProcess()

{

    int i;

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

    printf("    ProcessID    |    ProcessName    |    ProcessPriority    |    Running          \n");

    //进程号，进程名，优先级，是否在运行

    printf("-------------------------------------------------------------------------------\n");

    for (i = 0; i < NR\_TASKS + NR\_PROCS; i++)//逐个遍历

    {

        printf("        %d", proc\_table[i].pid);

        printf("                 %5s", proc\_table[i].name);

        printf("                   %2d", proc\_table[i].priority);

        if (proc\_table[i].priority == 0)

        {

            printf("                   no\n");

        }

        else

        {

            printf("                   yes\n");

        }

    }

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

}

int getMag(int n)

{

    int mag = 1;

    for (int i = 0; i < n; i++)

    {

        mag = mag \* 10;

    }

    return mag;

}

//计算进程pid

int getPid(char str[])

{

    int length = 0;

    for (; length < MAX\_FILENAME\_LEN; length++)

    {

        if (str[length] == '\0')

        {

            break;

        }

    }

    int pid = 0;

    for (int i = 0; i < length; i++)

    {

        if (str[i] - '0' > -1 && str[i] - '9' < 1)

        {

            pid = pid + (str[i] + 1 - '1') \* getMag(length - 1 - i);

        }

        else

        {

            pid = -1;

            break;

        }

    }

    return pid;

}

//结束进程

void killProcess(char str[])

{

    int pid = getPid(str);

    //健壮性处理以及结束进程

    if (pid >= NR\_TASKS + NR\_PROCS || pid < 0)

    {

        printf("The pid exceeded the range\n");

    }

    else if (pid < NR\_TASKS)

    {

        printf("System tasks cannot be killed.\n");

    }

    else if (proc\_table[pid].priority == 0 || proc\_table[pid].p\_flags == -1)

    {

        printf("Process not found.\n");

    }

    else if (pid == 4 || pid == 6)

    {

        printf("This process cannot be killed.\n");

    }

    else

    {

        proc\_table[pid].priority = 0;

        proc\_table[pid].p\_flags = -1;

        printf("Target process killed.\n");

    }

    showProcess();

}

//重启进程

void restartProcess(char str[])

{

    int pid = getPid(str);

    if (pid >= NR\_TASKS + NR\_PROCS || pid < 0)

    {

        printf("The pid exceeded the range\n");

    }

    else if (proc\_table[pid].p\_flags != -1)

    {

        printf("This process is already running.\n");

    }

    else

    {

        proc\_table[pid].priority = 1;

        proc\_table[pid].p\_flags = 1;

        printf("Target process is running.\n");

    }

    showProcess();

}

## 3.4 系统级应用——文件管理

主要函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 返回值类型 | 函数参数 | 作用 |
| WriteDisk | void | int len | 将文件存入硬盘 |
| ReadDisk | int | 无 | 读取硬盘内容 |
| CreateFile | int | char\* filename, int filetype | 创建文件 |
| ShowFileList | void | 无 | 展示当前这一级的文件 |
| SearchFile | int | char\* name | 搜索该目录下文件 |
| ReturnFile | void | int id | 返回上一级文件夹的id |
| DeleteFile | int | int id | 删除特定id的文件 |
| ShowMessage | void | 无 | 输出提示信息 |

核心代码

//文件管理主函数

void runFileManage(int fd\_stdin) {

char rdbuf[128];

char cmd[8];

char filename[120];

char buf[1024];

int m, n;

char \_name[MAX\_FILE\_NAME\_LENGTH];

FSInit();

int len = ReadDisk();

ShowMessage();

while (1) {

for (int i = 0; i <= 127; i++)

rdbuf[i] = '\0';

for (int i = 0; i < MAX\_FILE\_NAME\_LENGTH; i++)

\_name[i] = '\0';

printf("\n/%s:", blocks[currentFileID].fileName);

int r = read(fd\_stdin, rdbuf, 70);

rdbuf[r] = 0;

assert(fd\_stdin == 0);

char target[10];

for (int i = 0; i <= 1 && i < r; i++) {

target[i] = rdbuf[i];

}

//mkf指令定义make file

if (rdbuf[0] == 'm' && rdbuf[1] == 'k' && rdbuf[2] == 'f' ) {

if (rdbuf[3] != ' ') {

printf("You should add the filename, like \"create XXX\".\n");

printf("Please input [help] to know more.\n");

continue;

}

for (int i = 0; i < MAX\_FILE\_NAME\_LENGTH && i < r - 3; i++) {

\_name[i] = rdbuf[i + 4];

}

CreateFIle(\_name, 0);

}

//mkdir指令定义make directory

else if (rdbuf[0] == 'm' && rdbuf[1] == 'k' && rdbuf[2] == 'd' && rdbuf[3] == 'i' && rdbuf[4] == 'r') {

if (rdbuf[5] != ' ') {

printf("You should add the dirname, like \"mkdir XXX\".\n");

printf("Please input [help] to know more.\n");

continue;

}

char N[MAX\_FILE\_NAME\_LENGTH];

for (int i = 0; i < MAX\_FILE\_NAME\_LENGTH && i < r - 3; i++) {

\_name[i] = rdbuf[i + 6];

}

CreateFIle(\_name, 1);

}

//ls指令定义list

else if (rdbuf[0] == 'l' && rdbuf[1] == 's') {

showFileList();

}

//cd指令定义change directory

else if (rdbuf[0] == 'c' && rdbuf[1] == 'd') {

if (rdbuf[2] == ' ' && rdbuf[3] == '.' && rdbuf[4] == '.') {

ReturnFile(currentFileID);

continue;

}

else if (rdbuf[2] != ' ') {

printf("You should add the dirname, like \"cd XXX\".\n");

printf("Please input [help] to know more.\n");

continue;

}

for (int i = 0; i < MAX\_FILE\_NAME\_LENGTH && i < r - 3; i++) {

\_name[i] = rdbuf[i + 3];

}

printf("name: %s\n", \_name);

int ID = SearchFile(\_name);

if (ID >= 0) {

if (blocks[ID].fileType == 1) {

currentFileID = ID;

continue;

}

else if (blocks[ID].fileType == 0) {

while (1) {

printf("input the character representing the method you want to operate:"

"\nu --- update"

"\nd --- detail of the content"

"\nq --- quit\n");

int r = read(fd\_stdin, rdbuf, 70);

rdbuf[r] = 0;

if (strcmp(rdbuf, "u") == 0) {

printf("input the text you want to write:\n");

int r = read(fd\_stdin, blocks[ID].content, MAX\_CONTENT\_);

blocks[ID].content[r] = 0;

}

else if (strcmp(rdbuf, "d") == 0) {

printf("--------------------------------------------"

"\n%s\n-------------------------------------\n", blocks[ID].content);

}

else if (strcmp(rdbuf, "q") == 0) {

printf("would you like to save the change? y/n");

int r = read(fd\_stdin, rdbuf, 70);

rdbuf[r] = 0;

if (strcmp(rdbuf, "y") == 0) {

printf("save changes!");

}

else {

printf("quit without changing");

}

break;

}

}

}

}

else

printf("No such file!");

}

//rm指令定义remove

else if (rdbuf[0] == 'r' && rdbuf[1] == 'm') {

if (rdbuf[2] != ' ') {

printf("You should add the filename or dirname, like \"rm XXX\".\n");

printf("Please input [help] to know more.\n");

continue;

}

for (int i = 0; i < MAX\_FILE\_NAME\_LENGTH && i < r - 3; i++) {

\_name[i] = rdbuf[i + 3];

}

int ID = SearchFile(\_name);

if (ID >= 0) {

printf("Delete successfully!\n");

DeleteFile(ID);

for (int i = 0; i < blocks[currentFileID].childrenNumber; i++) {

if (ID == blocks[currentFileID].children[i]) {

for (int j = i + 1; j < blocks[currentFileID].childrenNumber; j++) {

blocks[currentFileID].children[i] = blocks[currentFileID].children[j];

}

blocks[currentFileID].childrenNumber--;

break;

}

}

}

else

printf("No such file!\n");

}

//sv指令定义save

else if (rdbuf[0] == 's' && rdbuf[1] == 'v') {

WriteDisk(1000);

printf("Save to disk successfully!\n");

}

//help指令定义

else if (strcmp(rdbuf, "help") == 0) {

printf("\n");

ShowMessage();

}

//quit指令定义

else if (strcmp(rdbuf, "quit") == 0) {

clear();

break;

}

//clear指令定义

else if (!strcmp(rdbuf, "clear")) {

clear();

}

//其他指令

else {

file\_usage();

printf("Sorry, there no such command in the File Manager.\n");

printf("You can input [help] to know more.\n");

}

## 3.5 用户级应用——日历

主要函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 返回值类型 | 函数参数 | 作用 |
| cal\_usage | void | void | 输出cal应用功能提示 |
| zeller | int | int year, int month, int day | 计算该日期的星期 |
| leap | int | int year | 判断该年份是否为闰年 |
| year | void | int year | 输出该年份年历 |
| month | void | int year, int month | 输出该月份月历 |
| day | void | int year, int month, int day | 输出该日期在该年位置和星期 |
| calMain | void | char\* opt | 主函数 |

核心代码

year

/\* 打印年历 \*/

void year(int y) {

int m, d, p1, i, p, sum = 0, s, mp;

int a[23] = { 0 }, b[13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 };

b[2] = 28 + leap(y);

printf("Calendar for %d : \n\n", y);

for (s = 0; s <= 3; s++) {

switch (s) {

case 0:

printf(" Jan Feb Mar\n");

break;

case 1:

printf(" Apr May Jun\n");

break;

case 2:

printf(" Jul Aug Sept\n");

break;

case 3:

printf(" Oct Nov Dec\n");

break;

default:

;

}

printf("Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa\n");

for (m = 3 \* s + 1; m <= 3 \* s + 3; m++) {

p1 = zeller(y, m, 1) + 8 \* (m - 3 \* s - 1);

for (d = 1; p1 % 8 - 1 + d <= 6; d++) {

a[p1 - 1 + d] = d;

}

}

for (i = 0; i < 23; i++) {

if (a[i] == 0) {

printf(" ");

}

else {

if (a[i] < 10) {

printf("%d ", a[i]);

}

else {

printf("%d ", a[i]);

}

}

}

printf("\n");

while (1) {

sum = 0;

for (mp = 1; mp <= 3; mp++) {

if (a[8 \* (mp - 1) + 6] == 0 || a[8 \* (mp - 1) + 6] == b[mp + 3 \* s]) {

for (p = 8 \* (mp - 1); p <= 8 \* (mp - 1) + 6; p++) {

a[p] = 0;

}

}

else {

a[8 \* (mp - 1)] = a[8 \* (mp - 1) + 6] + 1;

for (p = 8 \* (mp - 1); p <= 8 \* (mp - 1) + 6; p++) {

a[p] = p - 8 \* (mp - 1) + a[8 \* (mp - 1)];

if (a[p] > b[mp + 3 \* s]) {

a[p] = 0;

}

}

}

}

for (i = 0; i < 23; i++) {

sum = sum + a[i];

}

if (sum == 0) {

printf("\n");

break;

}

for (i = 0; i < 23; i++) {

if (a[i] == 0) {

printf(" ");

}

else {

if (a[i] < 10) {

printf("%d ", a[i]);

}

else {

printf("%d ", a[i]);

}

}

}

printf("\n");

}

}

printf("\n");

}

calMain

/\* 计算器主函数 \*/

void calMain(char\* option)

{

int b[13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 };

/\* 计算器命令行判断 \*/

char str\_list[3][10] = { "--year","--month","--day" };

int flag[3] = { 1,1,1 };

for (int i = 0; i < 3; ++i) {

int j = 0;

while (option[j] != ' ' && option[j] != '\0') {

if (option[j] != str\_list[i][j]) {

flag[i] = 0;

break;

}

++j;

}

}

int y, m, d;

char year\_str[5] = "\0", month\_str[3] = "\0", day\_str[3] = "\0";

if (!strcmp(option, "NULL")) {

cal\_usage();

}

else if (flag[0]) { //--year

if (strlen(option) > 7) {

char\* value = option + 7;

int i = 0;

for (int j = 0; i < strlen(value) && value[i] != '/' && value[i] != ' '; ++i, ++j) {

year\_str[i] = value[i];

}

atoi(year\_str, &y);

if (y > 0) {

year(y);

}

else {

if (y < 0) {

printf("ERROR:<year> should be greater than 0.\n");

}

printf("Please input again.\n");

}

}

else {

printf("Sorry, you should add [year].\n");

printf("You can input [cal] to know more.\n");

}

}

else if (flag[1]) { //--month

if (strlen(option) > 8) {

char\* value = option + 8;

int i = 0;

for (int j = 0; i < strlen(value) && value[i] != '/' && value[i] != ' '; ++i, ++j) {

year\_str[i] = value[i];

}

++i;

for (int j = 0; i < strlen(value) && value[i] != '/' && value[i] != ' '; ++i, ++j) {

month\_str[j] = value[i];

}

atoi(year\_str, &y);

atoi(month\_str, &m);

if (y > 0 && m > 0 && m < 13) {

month(y, m);

}

else {

if (y < 0) {

printf("ERROR:<year> should be greater than 0.\n");

}

if (m < 1 || m > 12) {

printf("ERROR:<month> should be between 1 to 12.\n");

}

printf("Please input again.\n");

}

}

else {

printf("Sorry, you should add [year/month].\n");

printf("You can input <cal> to know more.\n");

}

}

else if (flag[2]) { //--day

if (strlen(option) > 6) {

char\* value = option + 6;

int i = 0;

for (int j = 0; i < strlen(value) && value[i] != '/' && value[i] != ' '; ++i, ++j) {

year\_str[i] = value[i];

}

++i;

for (int j = 0; i < strlen(value) && value[i] != '/' && value[i] != ' '; ++i, ++j) {

month\_str[j] = value[i];

}

++i;

for (int j = 0; i < strlen(value) && value[i] != ' '; ++i, ++j) {

day\_str[j] = value[i];

}

atoi(year\_str, &y);

atoi(month\_str, &m);

atoi(day\_str, &d);

if (y < 0) {

printf("ERROR:<year> should be greater than 0.\n");

printf("Please input again.\n");

}

else if (m < 1 || m > 12) {

printf("ERROR:<month> should between 1 to 12.\n");

printf("Please input again.\n");

}

else if (d < 1 || d > b[m] + leap(y)) {

printf("ERROR:<day> should be legal.\n");

printf("Please input again.\n");

}

else{

day(y, m, d);

}

}

else {

printf("Sorry, you should add [year/month/day].\n");

printf("You can input [cal] to know more.\n");

}

}

else {

printf("Sorry, there no such option for cal.\n");

printf("You can input <cal> to know more.\n");

}

printf("\n");

return 0;

}

## 3.6 用户级应用——计算器

主要函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 返回值类型 | 函数参数 | 作用 |
| calcMain | void | char\* | 输出calc应用功能提示 |
| calculate | int | char\*,bool | 计算控制函数 |
| showBackMode | void | Data[], int | 输出后缀表达式 |
| NextContent | Data | char\* | 获取表达式的下一项 |
| calResult | int | Data[], int | 计算后缀表达式的结果 |
| Cal | int | int,char,int | 基本运算函数 |
| check\_exp\_bucket | bool | char\* | 判断表达式括号是否匹配 |
| check\_exp\_notion | bool | char\* | 判断表达式是否有非法符号 |

核心代码：

int Cal(int left, char op, int right)

{

    switch (op)

    {

    case '+':

        return left + right;

        break;

    case '-':

        return left - right;

        break;

    case '\*':

        return left \* right;

        break;

    case '/':

        return left / right;

        break;

    default:

        return left + right;

        break;

    }

}

void showBackMode(struct Data result[], int size) {

    printf("The reverse polish notation is: ");

    for (int i = 0; i < size; ++i) {

        if (result[i].flag == 0) {

            printf("%c ", result[i].data);

        }

        else {

            printf("%d ", result[i].data);

        }

    }

    printf("\n");

}

int calResult(struct Data result[], int size)

{

    num\_stack\_clear();

    for (int i = 0; i < size; ++i) {

        if (result[i].flag == 1) {

            num\_stack\_push(result[i].data);

        }

        else {

            int right = num\_stack\_pop();

            int left = num\_stack\_pop();

            num\_stack\_push(Cal(left, result[i].data, right));

        }

    }

    return num\_stack\_pop();

}

/\*初始两个栈\*/

    num\_stack\_clear();

    op\_stack\_clear();

    \_current = 0;

    struct Data result[100];

    int index = 0;

    /\*在表达式尾部添加结束标识符\*/

    addTail(exp);

    op\_stack\_push('#');

    struct Data elem = NextContent(exp);

    while (!isempty\_op\_stack()) {

        char ch = elem.data;

        if (elem.flag == 1) {       //如果是操作数, 直接读入下一个内容

            result[index] = elem;

            index++;

            elem = NextContent(exp);

        }

        else if (elem.flag == 0) {

            char topch = op\_stack\_top();

            if (isp(topch) < icp(ch)) {     /

                op\_stack\_push(ch);

                elem = NextContent(exp);

            }

            else if (isp(topch) > icp(ch)) {    //当前优先级小,推展并输出到结果中

                struct Data buf;

                buf.data = op\_stack\_pop();

                buf.flag = 0;

                result[index] = buf;

                index++;

            }

            else {

                if (op\_stack\_top() == '(') {

                    elem = NextContent(exp);

                }

                op\_stack\_pop();

            }

        }

    }

    if (if\_showrev) {

        showBackMode(result, index);

    }

    return calResult(result, index);

}

/\*判断表达式括号是否匹配\*/

bool check\_exp\_bucket(char\* exp) {

    char ch = '\0';

    for (int i = 0; i < strlen(exp); ++i) {

        if (exp[i] == '(') {

            bucket\_stack\_push('(');

        }

        else if (exp[i] == ')') {

            ch = bucket\_stack\_pop();

            if (ch == EMPTY\_CH || ch != '(') {

                printf("Buckets in the exprssion you input do not match.\n");

                return False;

            }

        }

    }

    return isempty\_bucket\_stack();

}

/\*判断表达式括号是否匹配\*/

bool check\_exp\_bucket(char\* exp) {

    char ch = '\0';

    for (int i = 0; i < strlen(exp); ++i) {

        if (exp[i] == '(') {

            bucket\_stack\_push('(');

        }

        else if (exp[i] == ')') {

            ch = bucket\_stack\_pop();

            if (ch == EMPTY\_CH || ch != '(') {

                printf("Buckets in the exprssion you input do not match.\n");

                return False;

            }

        }

    }

    return isempty\_bucket\_stack();

}

/\*判断表达式是否有非法符号\*/

bool check\_exp\_notion(char\* exp) {

    for (int i = 0; i < strlen(exp); ++i) {

        if (isDigit(exp[i]) || isOperator(exp[i]) || exp[i] == ' ') {

            continue;

        }

        else {

            printf("The operator we support: [+-\*/()], you have input %c.\n", exp[i]);

            return False;

        }

    }

    return True;

}

/\*判断表达式是否有非法符号\*/

bool check\_exp\_notion(char\* exp) {

    for (int i = 0; i < strlen(exp); ++i) {

        if (isDigit(exp[i]) || isOperator(exp[i]) || exp[i] == ' ') {

            continue;

        }

        else {

            printf("The operator we support: [+-\*/()], you have input %c.\n", exp[i]);

            return False;

        }

    }

    return True;

}

## 3.7 用户级应用——2048

主要函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 返回值类型 | 函数参数 | 作用 |
| GAME2048 | void | int fd\_stdin, int fd\_stdout | 2048游戏主函数 |
| loop\_game | void | int fd\_stdin | 循环进行游戏 |
| reset\_game | void | 无 | 重置 |
| release\_game | void | int signal | 游戏结束，清空指令 |
| check\_game\_over | void | 无 | 判断游戏是否结束 |
| move\_left | void | 无 | 向左移动 |
| move\_right | void | 无 | 向右移动 |
| move\_up | void | 无 | 向上移动 |
| move\_down | void | 无 | 向下移动 |
| refresh\_screen | void | 无 | 刷新界面 |

核心代码

/\*游戏2048主函数 \*/

void GAME2048(fd\_stdin, fd\_stdout)

{

clear();

init\_game();

loop\_game(fd\_stdin);

release\_game(0);

}

/\*循环游戏 \*/

void loop\_game(int fd\_stdin) {

while (1) {

char rdbuf[128];

int r = 0;

r = read(fd\_stdin, rdbuf, 70);

if (r > 1)

{

refresh\_screen();

continue;

}

rdbuf[r] = 0;

char cmd = rdbuf[0];

//判断是否退出游戏

if (if\_prepare\_exit) {

if (cmd == 'y' || cmd == 'Y') {

clear\_screen();

//清屏后退出游戏

return;

}

else if (cmd == 'n' || cmd == 'N') {

if\_prepare\_exit = 0;

refresh\_screen();//取消退出

continue;

}

else {

continue;

}

}

//判断是否已经输掉游戏

if (if\_game\_over) {

if (cmd == 'y' || cmd == 'Y') {

reset\_game();

continue;

}

else if (cmd == 'n' || cmd == 'N') {

clear();//取消

return;

}

else continue;

}

if\_need\_add\_num = 0;//先设定不默认需要生成随机数，需要时再设定为1

//指令判断，wsad代表上下左右命令，q代表退出

switch (cmd) {

case 'a':

move\_left();

break;

case 's':

move\_down();

break;

case 'w':

move\_up();

break;

case 'd':

move\_right();

break;

case 'q':

if\_prepare\_exit = 1;

break;

default:

refresh\_screen();

continue;

}

//生成随机数同时刷新显示，

if (if\_need\_add\_num) {

add\_rand\_num();

refresh\_screen();

}

else if (if\_prepare\_exit) {

refresh\_screen();

}

}

}