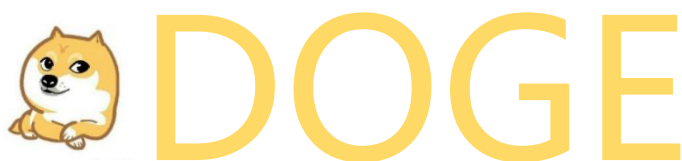


操作系统课程设计文档



Operating System

同济大学软件学院

R&D TEAM

1852448	李源峰
1851489	吕梓源
1854139	周子怡
1852342	何娴
1853316	杜玫

2020.8.20

目 录

1 项目概述.....	4
1.1 环境配置	4
1.2 项目简介	4
2 项目运行流程.....	4
2.1 系统运行流程	4
2.2 boot.bin	4
2.3 loader.bin	5
2.4 保护模式	5
2.5 kernal.bin	5
3 开发成果.....	5
3.1 完成 64 位到 32 位移植.....	5
3.2 增强对 C 语言开发的支持.....	5
3.3 开机动画	6
3.4 主控制台及游戏广场	6
3.5 用户级程序	8
3.5.1 日历	8
3.5.2 推箱子	9
3.5.3 猜数字	10
3.5.4. 计算器	10
3.5.5. 五子棋	11
3.5.6. 扫雷	13
3.5.7. 2048 游戏	14
3.5.8 石头剪刀布	16
3.5.9 计时器 1	17
3.5.10. 计时器 2	19
3.6. 文件系统	19
3.6.1 基本功能	19
3.6.2 实现思路	20
3.6.3 主要变量简介	20
3.6.4 功能函数及简介	20

3.6.5 截屏示例	21
3.7. 进程管理系统	25
3.7.1 基本功能	25
3.7.2 函数实现	26
3.7.3 截屏示例	26
4. 成员分工	27
5. 项目开发历程	28

1 项目概述

1.1 环境配置

编写语言：汇编语言、C 语言

开发环境：Ubuntu20.4

运行环境：Bochs2.6.9

1.2 项目简介

(1).本项目参照《Orange's: 一个操作系统的实现》中示例代码，完成搭建一个基础的操作系统，并在此基础上对多个模块进行了修改；与操作系统内核多次交互，通过调用较多的系统 API 实现对系统的应用。

(2).主要完成的功能：

A)Orange' s 操作系统原有的功能

B)增强了对 C 语言开发的支持

C)多个用户应用及系统应用

E)文件管理系统

F) 进程管理系统

2 项目运行流程

2.1 系统运行流程

概述：引导→加载内核入内存→跳入保护模式→开始执行内核

详细过程：从软盘引导 → 在软盘中查找 Loader.bin → 加载 Loader.bin → 跳转至 Loader.bin 中的代码开始执行 → 在软盘中查找系统内核 kernel.bin → 进入保护模式 → 加载 kernel.bin → 跳转至 kernel.bin 中的代码 开始执行 → 更新 GDT → 初始化 TSS → 跳入系统主函数 → 启动系统进程 → 开启时钟中断 → 开始进程调度 → (系统开始运转)

2.2 boot. bin

DOGE 的启动盘是虚拟软盘 a.img，其文件系统格式为 FAT12。在 BOOT 开始时，会加载 BootSector 到内存 0x7c00，并调用 BIOSINT13 中断复位软驱。之后，系统将在 A 盘根目录区查找 loader.bin，若未找到，系统将提示 未找到 loader.bin，并陷入死循环。若找到，系统将加载 loader.bin，并开始执行。我们选择将启动系统、加载 kernel 的过程放入

loader.bin，是因为 bootsector 大小有限，仅 512B，当要做的工作很多时，bootsector 可能会不够用。但是使用 loader 就不受此限制了。

2.3 loader.bin

loader.bin 开始时，首先会调用 BIOS INT15 中断，获取内存信息，为后续启动分页机制做准备。之后再次调用 BIOSINT13 中断复位软驱，并在 A 盘根目录区查找 kernel.bin，查找不到则提示未找到 KERNEL.BIN，找到则关闭软驱马达并进入保护模式。

2.4 保护模式

loader.bin 开始时，首先会调用 BIOS INT15 中断，获取内存信息，为后续启动分页机制做准备。之后再次调用 BIOSINT13 中断复位软驱，并在 A 盘根目录区查找 kernel.bin，查找不到则提示未找到 KERNEL.BIN，找到则关闭软驱马达并进入保护模式。

2.5 kernal.bin

进入 kernel 后，首先更新 GDT，之后初始化 TTS，进入系统主程序。在主程序中，先初始化系统任务 and 用户进程，之后开启时钟中断，进行进程调度，开启键盘中断。当准备工作都结束后，系统将跳出 RING0，开始运作

3 开发成果

3.1 完成 64 位到 32 位移植

由于 Orange's 是在 32 位的 Ubuntu 开发，而现在的 Ubuntu 均为 64 位，所以导致书上样例代码无法通过编译。在开发过程中，我们在 makefile 文件中加入了 gcc -m32 以及 ld -m elf_i386，使得编译器可以在 64 位的系统下编译出 32 位的机器代码。同时，我们还通过 -fno-stack-protector 禁用了 linux 的堆栈保护措施，成功解决了样例代码与我们的环境不兼容的问题。

3.2 增强对 C 语言开发的支持

(1).在 Orange's 操作系统实例中，操作系统的实现是由汇编语言和 C 语言共同实现的，但实际开发过程中，由于准备时间有限，对汇编及硬件知识掌握程度要求较高，我们编写了多个函数增强了 Orange's 对 C 语言开发的支持，并完全使用 C 语言对应用程序进行开发。

编写的函数包括:

PUBLIC void clear();	实现对屏幕的清空
PUBLIC int atoi(const char*src);	将字符串转换为整型数（支持正负数）
PUBLIC int toStr3(int i);	将整型数转换为 3 位字符串
PUBLIC int toInt(char *temp);	将 3 位字符串转换为整型数

- (2). 在开发过程中,我们发现由于 Orange's 操作系统中对内存的设计过小,导致我们编写的函数增加时,bochs 便无法加载,会提示 too large 的错误。于是我们修改了 boot 文件夹中的 load.inc 文件,修改了其中的寄存器地址,完成了对内存的扩容。

3.3 开机动画

DOGE 设置了共 8 帧长约 3 秒的开机动画,该动画会在开机时自动播放:



3.4 主控制台及游戏广场

DOGE 开机后会自动进入主控制台,用户可以根据主控制台提供的指令进行操作:

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

=====
WELCOME DOGE OS
V 5.6
=====

#           MENU           #
#           #           #
# [COMMAND] [FUNCTION]    #
#           #           #
# $ calendar : Open calendar #
# $ timer1   : Open 1# timer  #
# $ timer2   : Open 2# timer  #
# $ calcu    : Open Calculator #
# $ game     : Go to game square #
# $ fs       : Open file system #
# $ pm       : Open process manager #
# $ clear    : Clear screen #
# $ help     : Show commands #
# $ about    : About DOGE #
#           #           #
# DOGE R&D TEAM #
# ALL RIGHT REVERSED #
#           #           #
=====

DOGE $
IPS: 84,119M  A: NUM CAPS SCRL HD:0-M
```

在主控制台输入 game 可进入游戏广场，用户可以在此选择游戏进入：

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

=====
0.0 DOGE 0.0
!!!GAME SQUARE!!!
=====

#           HAVE A GOOD TIME!           #
#           #           #
# [COMMAND] [GAME]                      #
#           #           #
# $ 1       : Guess number              #
# $ 2       : Push box                  #
# $ 3       : Five-in-a-row             #
# $ 4       : Mine clearance            #
# $ 5       : 2048 game                 #
# $ 6       : Rock-paper-scissors       #
# $ clear   : Clear screen              #
# $ help    : Show games                #
# $ quit    : exit GAME SQUARE         #
#           #           #
# DOGE R&D TEAM #
# ALL RIGHT REVERSED #
#           #           #
=====

DOGE GAME SQUARE $
IPS: 84,176M  A: NUM CAPS SCRL HD:0-M
```

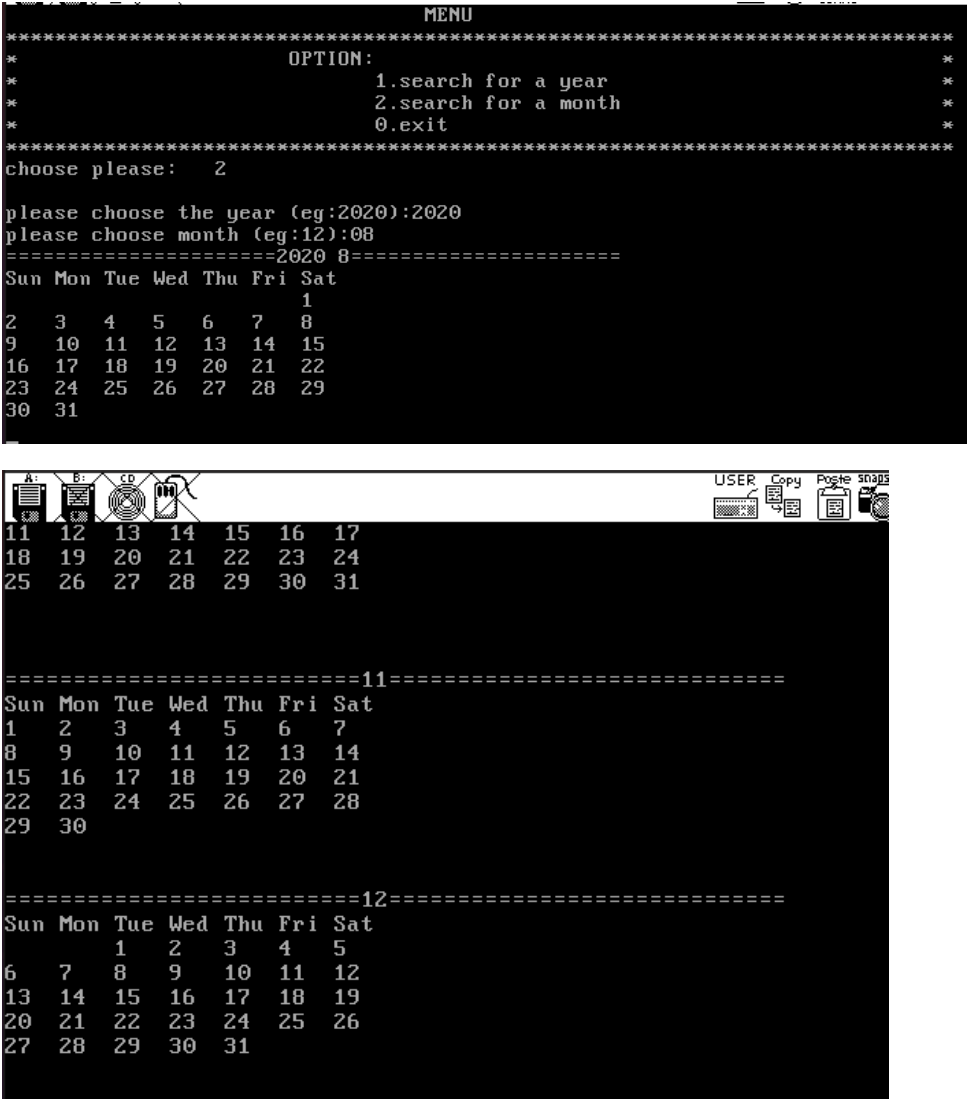
3.5 用户级程序

3.5.1 日历

3.5.1.1 使用说明

- (1).在主界面输入 calendar 进入日历
- (2).根据提示选择查看一年日历或查看一年中某月的日历，也可选择 0 退出日程序
- (3).根据提示输入需要查找的年份或月份
- (4).系统将在计算后返回对应的日历

3.5.1.2 效果实现



3.5.1.3 核心代码

- (1).闰年判断：通过使用公式检查用户输入年份是否为 400 的整数倍或是否为 4 的倍数来判断是否为闰年，若为闰年需要更改天数数组。
- (2).星期打印：使用公式 $w=(y+[y/4]+[c/4]-2c+[26(m+1)/10]+d-1)\%7$ 判断该月第一天为星期中的哪一天，便于后续打印。
- (3).年日历打印：使用 read 函数将用户输入读入，再通过公式得出正确年份，并根据年份对日历进行打印。

3.5.2 推箱子

3.5.2.1 使用情况

- (1).在主界面输入 game 进入游戏选择，输入 2 进入推箱子
- (2). 游戏中，“.”代表可走路径；“#”代表墙；“@”代表箱子；“0”代表箱子终点；“S”代表用户
- (3)在地图打印后，用户可以选择对人物进行上下左右方向的移动；
- (4).移动时若碰到墙壁则无法移动，若碰到箱子，则可以推着箱子前进；
- (5).当所有箱子都被移动到终点时游戏结束；
- (6).可以选择输入“q”结束游戏，回到游戏选择

3.5.2.2 效果实现



3.5.2.3 效果实现

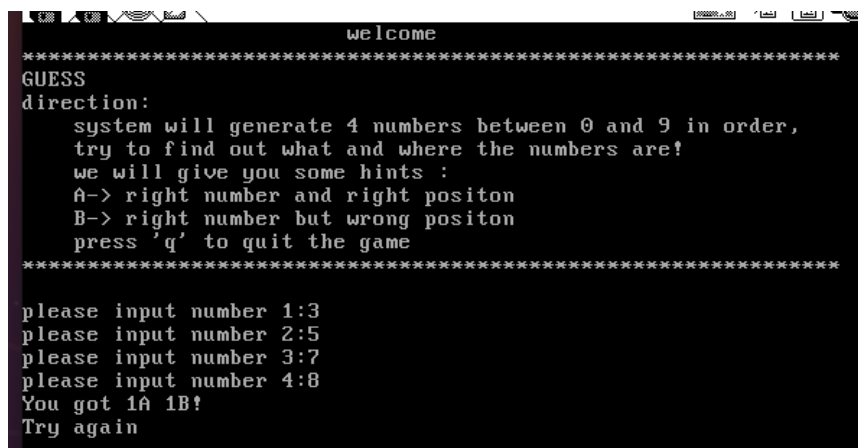
- (1).地图数组：使用一个 `int[7][8]` 数组记录地图中不同的标志以及用户位置
- (2).打印地图：根据地图数组记录的数数字，循环打印出路径、围墙、箱子、终点和用户
- (3).位置移动：再获得用户输入后判断该方向是否可以通过，若可以通过，则对数组进行更改，并判断更改后的数组是否实现箱子到达终点。若所有箱子均到大终点，则游戏结束

3.5.3 猜数字

3.5.3.1 使用说明

- (1).在主界面输入 `game` 进入游戏选择，输入 `1` 进入猜数字
- (2).游戏进行初始化，在游戏开始后根据提示依次按从小到大的顺序输入 `0~9` 间 `4` 个数字
- (3).在对比用户输入的数据后系统将对数字结果进行提示：
 - A 代表数字及其位置均正确；
 - B 代表数字存在于结果中，但位置不正确；
- (4).成功猜测所有数字后游戏胜利。或者输入“`q`”退出游戏。

3.5.3.2 效果实现



```
welcome
=====
GUESS
direction:
system will generate 4 numbers between 0 and 9 in order,
try to find out what and where the numbers are!
we will give you some hints :
A-> right number and right positon
B-> right number but wrong positon
press 'q' to quit the game
=====

please input number 1:3
please input number 2:5
please input number 3:7
please input number 4:8
You got 1A 1B?
Try again
```

3.5.3.2 核心代码

- (1).产生随机数字：通过使用 `get_ticks()`函数产生一个 `0~9` 之间随机数并验证该数字是否存在，若不存在则存入系统数组
- (2).正确数字检测：通过比较系统数组和用户输入数组，向用户反馈游戏结果

3.5.4. 计算器

3.5.4.1 使用说明

- (1).在主界面输入 `calcu` 进入计算器
- (2).根据提示，输入 `0` 退出游戏，回到游戏菜单，输入 `1` 进入游戏；在根据提示输入要计算的多项式，不用以“`=`”结尾

3.5.3.2 效果实现

```
-----
      input your choice
          0.exit
          1.enter
-----
1
#####
Please enter your polynomial
Enter q to back to exit
#####
3+5
Result 8
#####
Please enter your polynomial
Enter q to back to exit
#####
```

3.5.4.3 核心代码

- (1).处理用户多项式：如果是操作数，直接将 char 转化为数字并押入数字栈即可；如果是操作符，则要根据优先级加以判断和处理
- (2).设置优先级并且处理操作符：优先级的处理利用了栈内和栈外两个优先级
- (3).根据优先级处理操作符

3.5.5. 五子棋

3.5.5.1 使用说明

- (1).在控制台输入 game 进入游戏选择，输入 3 进入五子棋
- (2).根据提示，输入 0 退出游戏，回到游戏菜单，输入 1 进入游戏；在根据提示输入玩家落子的 x 坐标和 y 坐标，玩家落子后，清屏，显示玩家和电脑 AI 在这一局的落子

3.5.5.2 效果实现

```
-----
input your choice
0.exit
1.enter
-----
```

```
 1 2 3 4 5 6 7 8 9 10
1  | | | | | | | | | |
2  | | | | | | | | | |
3  | | | | | | | | | |
4  | | | | | | | | | |
5  | | | | | | | | | |
6  | | | | | | | | | |
7  | | | | | | | | | |
8  | | | | | | | | | |
9  | | | | | | | | | |
10 | | | | | | | | | |
Please input the line position where you put your Chess(x) and q to exit: _
```

```
 1 2 3 4 5 6 7 8 9 10
1  | | | | | | | | | |
2  | | | | | | | | | |
3  | | o | | | | | | |
4  | | | | | | | | | |
5  | | | | | | | | | |
6  | | | | | | | | | |
7  | | | | | | | | | |
8  | | | | | | | | | |
9  | | | | | | | | | |
10 | | | | | | | | | |
Please input the line position where you put your Chess(x) and q to exit: _
```

3.5.5.3 核心代码

(1).显示棋盘：在每次落子之后，此函数会首先清屏，再输出棋盘。棋子和棋盘都使用 ASCII 码对应的符号进行输出，最大限度模拟棋子的形状。

(2).此函数没有直接安排输出的数组，而是用一个 `chess_board[][]` 数组来标记哪些是用户、AI 和空白的数组，然后在后续的棋盘遍历中直接用 `printf()` 输出。(3 判断游戏是否结束：扫描整个棋盘，在有子的位置进行检查。首先检查四个方向是否可以赢，利用 `direction[][]` 这个数组检查四个方向能否获胜，由于是顺着棋盘扫描的，所以不需要检查上下，只需要检查“上”这一个单边方向即可。在扫描的时候，计数四个方向上同样颜色的棋子数，如果是 5 个，就结束游戏。

(3).AI 的估值函数：是用于判断棋子情况并且赋值的函数，可以帮助 AI 判断棋盘中棋子的情况，在八个方向上计算自己有多少相连的棋子，边上有多少墙或者对方的棋子，然后找到某一个方向上最大的连续情况。根据棋盘和五子棋的规则，给不同情况赋权重，有利于 AI 的棋力增强

(4).AI 落子函数：扫描整个棋盘，评估每个空白点的分数。分数是由防御+攻击价值组成的，分数越高，此点越好

3.5.6. 扫雷

3.6.6.1 使用说明

- (1).在控制台输入 game 进入游戏选择，输入 4 进入扫雷
- (2).根据提示，输入 0 退出游戏，回到游戏菜单，输入 1 进入游戏；在根据提示输入玩家落子的 x 坐标和 y 坐标，根据扫雷游戏的规则，显示可以被显示出的所有雷阵。

3.5.6.2 效果实现

```
-----  
input your choice  
0.exit  
1.enter  
-----
```

```
  1 2 3 4 5 6 7 8 9  
1 - - - - - - - -  
2 - - - - - - - -  
3 - - - - - - - -  
4 - - - - - - - -  
5 - - - - - - - -  
6 - - - - - - - -  
7 - - - - - - - -  
8 - - - - - - - -  
9 - - - - - - - -  
#####  
Please enter the x and y  
Enter the q to back to exit  
#####  
Please enter the x :
```

```
  1 2 3 4 5 6 7 8 9  
1 - - - - - - - -  
2 - - 2 - - - - -  
3 - - - - - - - -  
4 - - - - - 1 - -  
5 - - - - - - - -  
6 - - - - - - - -  
7 - - - - - - - -  
8 - - - - - - - -  
9 - - - - - - - -  
#####  
Please enter the x and y  
Enter the q to back to exit  
#####  
Please enter the x :_
```

3.5.6.3 核心代码

(1).扫雷游戏的实现中关键点之一就是要设置两个数组--mine 和 show，show 相当于是雷阵的盖子，也是我们用户在游戏结束之前一直看到的矩阵--加密矩阵

mine[][]:是真实的地雷阵，在其中，0 表示安全区域，1-8 表示此方块的上下左右以及 4 个斜对角线中地雷的数量，输入是\$则表示这个已经显示给用户了

show[][]:是用户看到的加密矩阵，没有被用户探测到的区域统一用“-”表示，被用户探索到的安全区域的区域就用“ ”表示，数字代表周围 8 个方向有几颗雷

(2).避免玩家第一次就踩雷：这个函数在玩家第一次选定了位置后再调用，即玩家第一次落子后再放置地雷，并且要求不能放在玩家第一次选定的位置处

(3).自动展开：此函数是典型的递归搜索问题，在上学期的算法课程中，我们有一道找单词的作业题，比此函数的逻辑更加复杂，但是主要思想都一样，即都是利用递归在每个方向搜索下去，直到遇到不符合条件的情况。在此函数中，我们要在 8 个方向上检验，跳出递归的条件是遇到不是'0'的情况，跳出递归后，再检验跳出的这个方块是不是数字，如果是数字，也可以展开。所以最后的效果就是，如果点击到了非雷非数字的方块，就会自动给展开它周围所有的非雷非数字方块以及这个区域最外圈是数字的方块

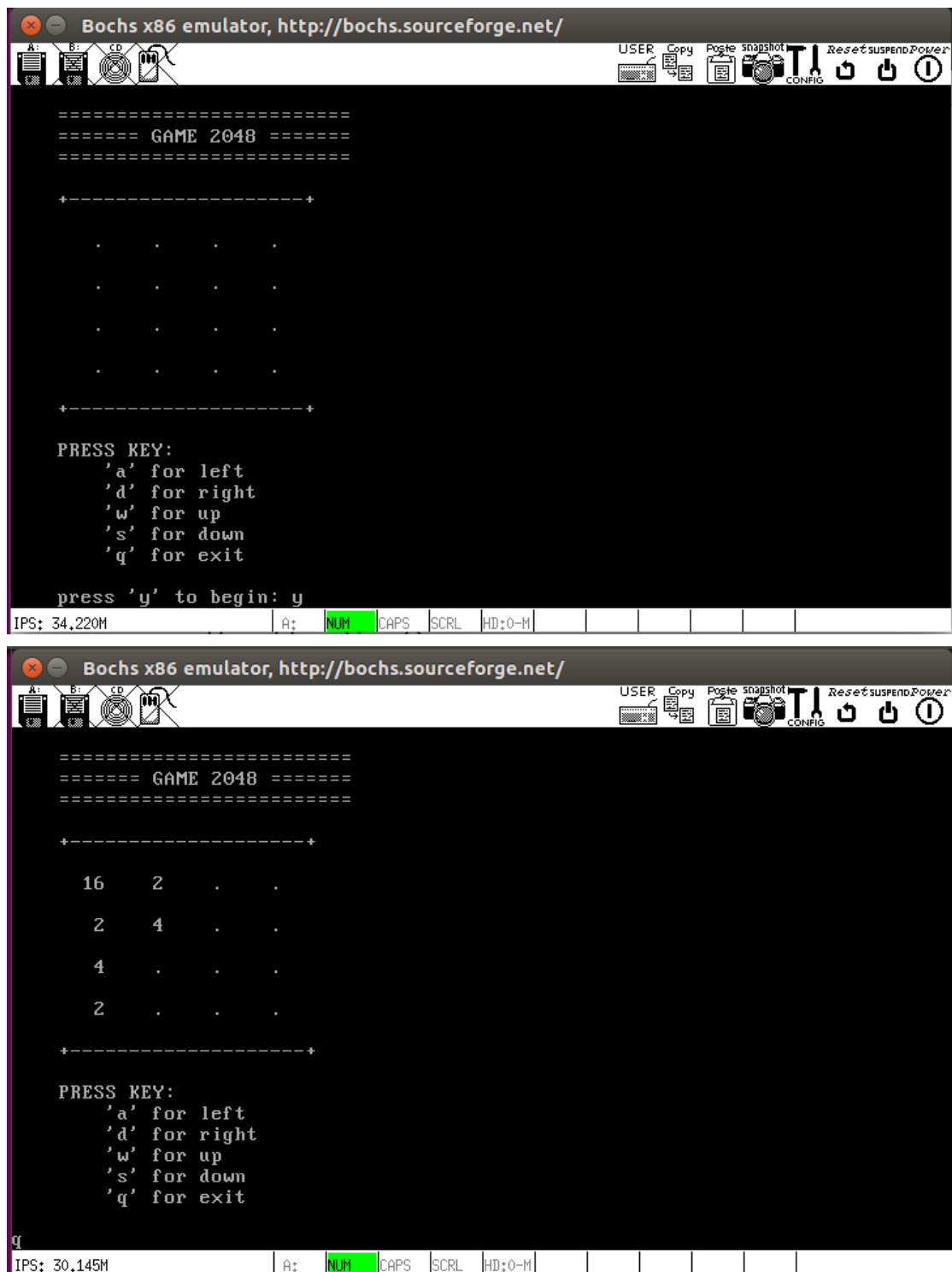
3.5.7. 2048 游戏

3.5.7.1 使用说明

(1).在控制台输入 game 进入游戏选择，输入 5 进入 2048 游戏

(2).操作步骤：通过方向键“wasd”进行游戏，按“q”退出游戏

3.5.7.2 效果实现



3.5.7.3 核心代码

(1) 2048 重要的是读取用户的键盘操作并且根据 2048 的游戏规则做出相应的反映，例如，按下左方向键后，`cur` 是当前处理列号，`change` 用于表示前一个数字是否合并。如果与前一个数相等且前一个数不是合并得来则与前一个数合并，否则靠左边紧密排列。

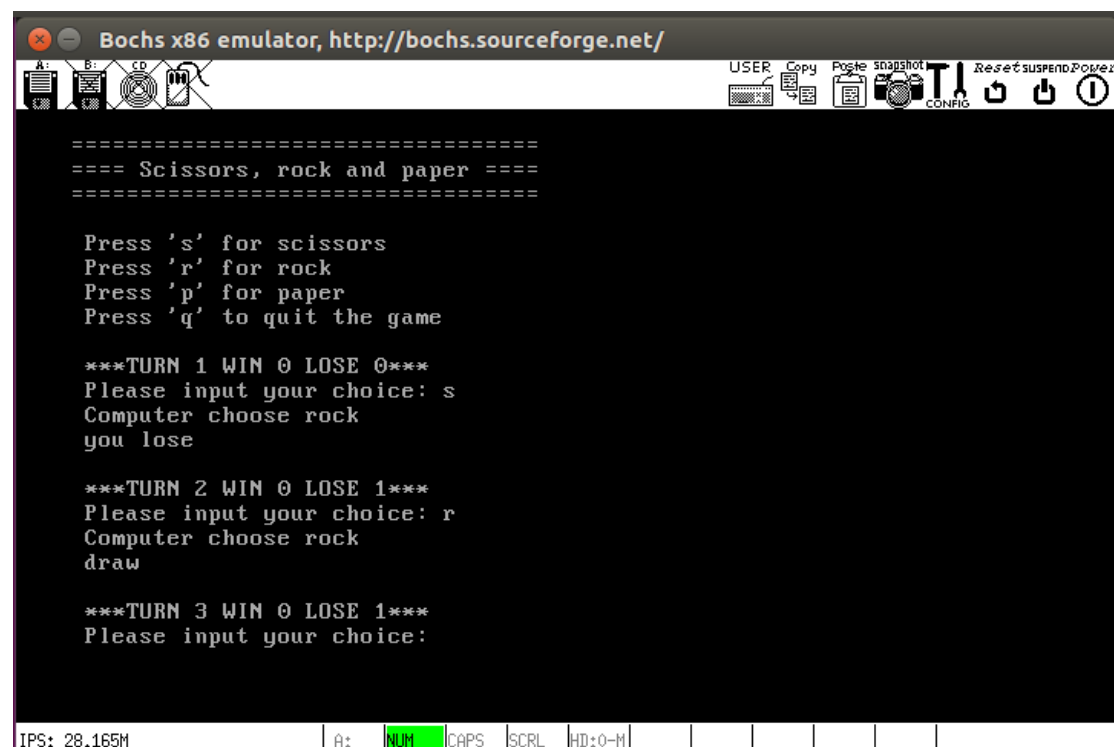
(2) 非阻塞输入函数：非阻塞意思为没有输入的时候不能一直等着，即 `READ()` 函数如果没有读取到用户输入，也要进行循环。出现阻塞的根源是 `keyboard.c` 中 `while` 等待键盘输入，首先我想到改 `while` 为 `if`，无输入时直接返回，不能得到想要的效果，后来试着写非阻塞输入函数 `getch()`，参考第 7 章 `write()` 系统调用实现 `printf()`--增加函数 `getch()`。但是由于进程结构改变所以出现了停滞现象,即将 `PROCESS` 改为 `struct proc`，进程对应 `tty` 获取方式改变了。

3.5.8 石头剪刀布

3.5.8.1 使用说明

- (1).在控制台输入 `game` 进入游戏选择，输入 6 进入石头剪刀布游戏
- (2).操作步骤：输入's'为剪刀，'r'为石头，'p'为布；电脑随机做出选择

3.5.8.2 效果实现



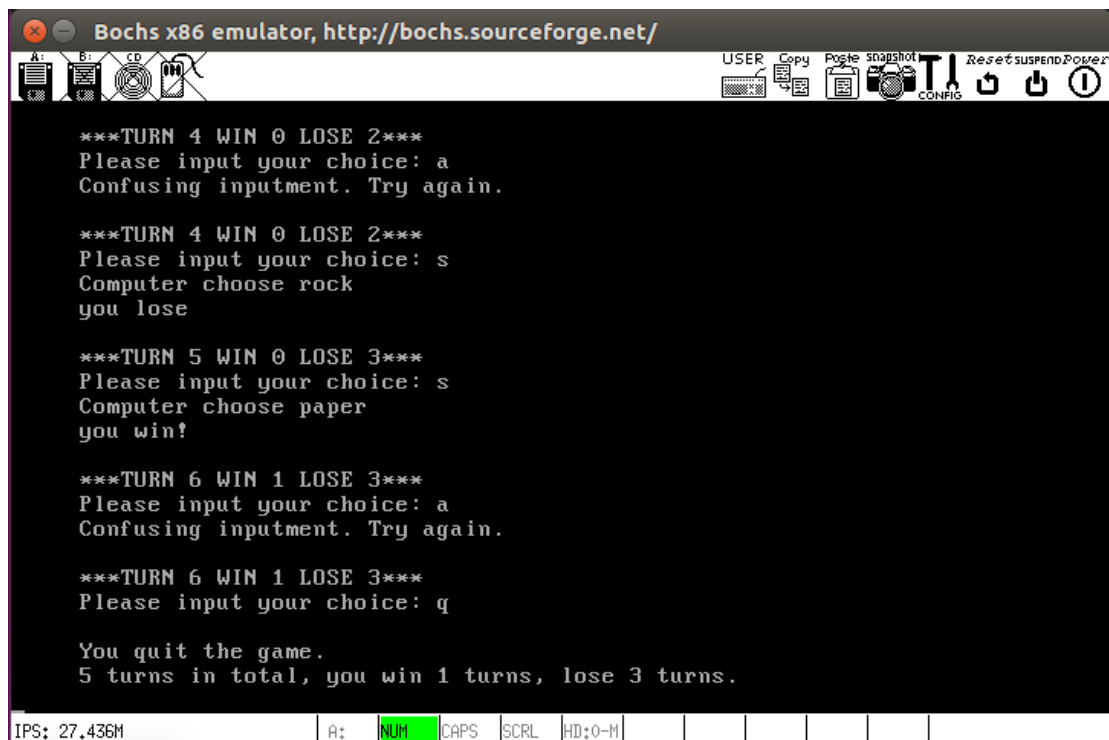
```
=====
=== Scissors, rock and paper ===
=====

Press 's' for scissors
Press 'r' for rock
Press 'p' for paper
Press 'q' to quit the game

***TURN 1 WIN 0 LOSE 0***
Please input your choice: s
Computer choose rock
you lose

***TURN 2 WIN 0 LOSE 1***
Please input your choice: r
Computer choose rock
draw

***TURN 3 WIN 0 LOSE 1***
Please input your choice:
```

3.5.8.3 核心代码

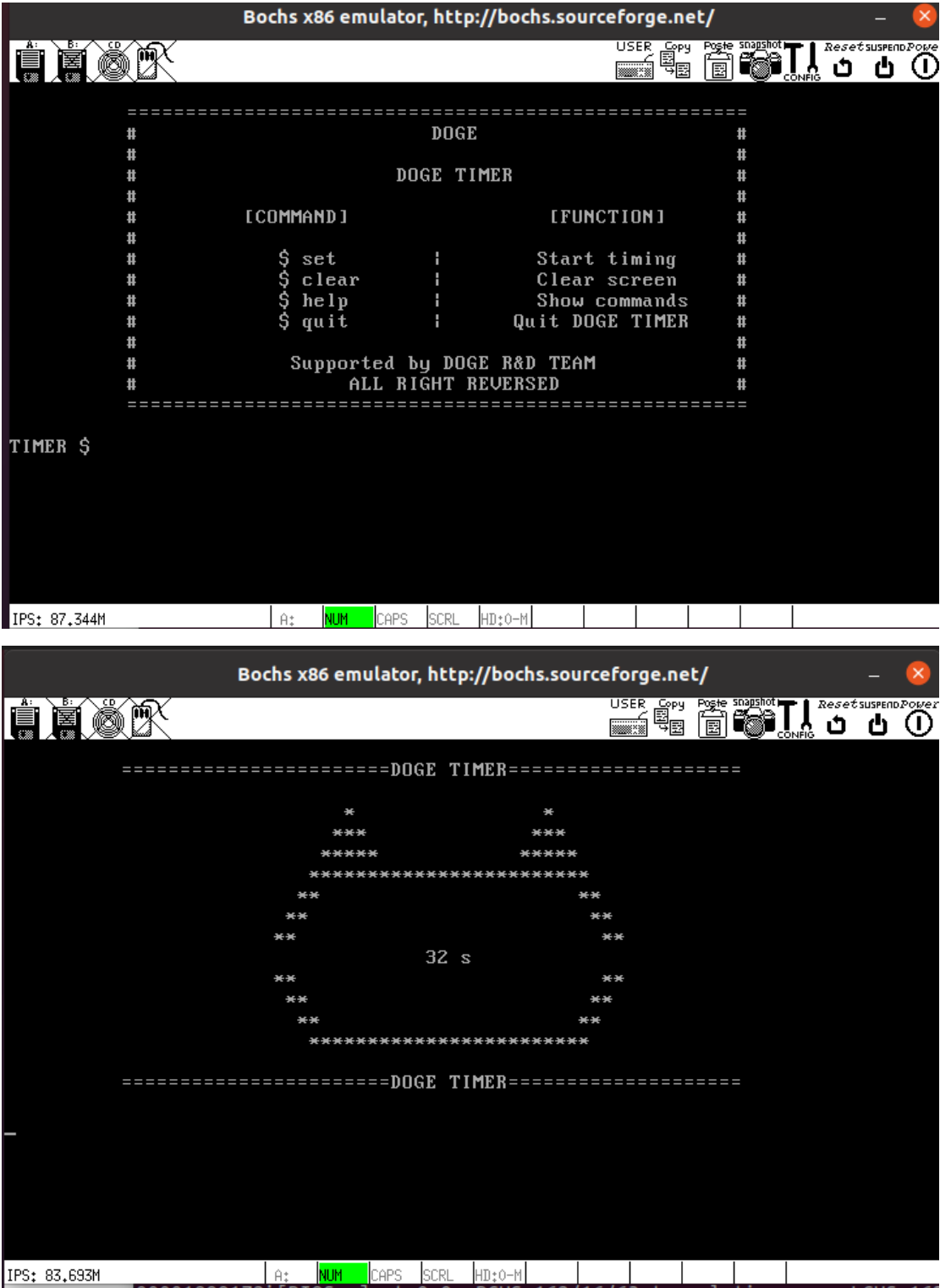
(1) 判断输赢：记剪刀为 0，石头为 1，布为 2 用户选择减去电脑随机选择的模，0 为平局，1 为用户胜，2 为电脑胜

3.5.9 计时器 1

3.5.9.1 使用说明

- (1).在主界面输入 timer1，进入定时器
- (2).输入 set，进入时间设置，输入秒数，计时器开始计时

3.5.9.2 效果实现



3.5.9.3 核心代码

利用在 string.h 中自己编写的 atoi 函数，提取输入的字符串，并将其转换为整型数，然

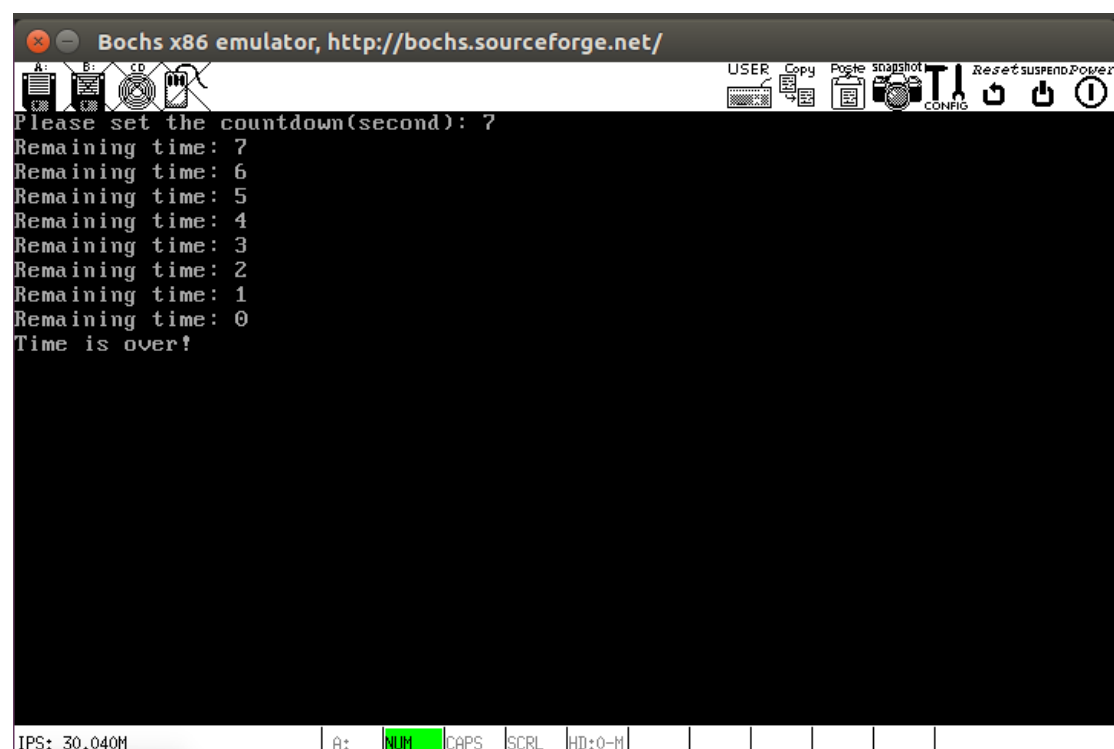
后通过 orange's 提供的 `milli_delay()`接口，通过 `for` 循环完成 1 秒一次刷新，完成计时。

3.5.10. 计时器 2

3.5.10.1 使用说明

- (1).在主界面输入 `timer1`，进入定时器
- (2).直接输入秒数即可让计时器开始计时

3.5.10.2 效果实现



3.5.9.3 核心代码

利用在 `string.h` 中自己编写的 `atoi` 函数，提取输入的字符串，并将其转换为整型数，然后通过 orange's 提供的 `milli_delay()`接口，通过 `for` 循环完成 1 秒一次刷新，完成计时。

3.6. 文件系统

3.6.1 基本功能

在 Orange 操作系统中，作者实现的是一个只支持打开文件、读取内容、写入内容、关闭文件的单级文件系统。DOGE 在基于 Orange 提供的框架模式下，将该文件系统改造成了

一个多级文件系统。DOGE 文件系统支持的功能不仅包括创建、删除、读写和查看已有文件，还包括创建文件夹，进入文件夹，展示当前文件夹下的所有文件目录等功能。只要用户在文件系统进行操作后输入\$ sv 指令进行写入硬盘操作，文件系统中的所有功能都会保存到虚拟硬盘 80m.img 中，在关机后，文件也会得到正确保存，再次开机仍能进行管理。

3.6.2 实现思路

在 Orange's 的文件相关接口的基础上，该系统通过一个固定的系统文件存储文件系统中的所有信息。在用户进入文件系统后，系统首先调用 orange's 提供的 open()接口创建一个系统文件（如果已经存在则是自动打开），然后再通过 read()接口将系统文件的内容读入内存中的 blocks[]结构体数组中。用户在进行操作时，实际操作的是内存中的 blocks[]结构体数组，如果用户输入了\$ sv 指令，那么文件系统便会调用 write()接口将 blocks[]的内容写入系统文件中。

在文件系统的实际工作过程中，每一个指令执行的工作都被封装成了相应的函数，当主函数接受到用户的输入后，首先会对输入的字符串进行分析，分析出相应的指令符，然后调用相应的函数实现相应的功能。如果用户的输入不是有效的指令，系统会给出报错信息，并等待用户的下一次输入。

DOGE 文件系统的所有函数均可在 kernal/main.c 中进行查阅。Orange's 提供的接口的函数定义可以在 lib/open 和 lib/write 查阅，其具体实现过程是在该函数内向硬驱（kernal/hd.c）发送消息，硬驱通过相应的消息类型调用 fs/中的函数进行与硬盘的交互。

3.6.3 主要变量简介

```
struct fBlock {
    int fID; //文件 ID
    char fName[MAX_FILE_NAME_LENGTH]; //文件名
    int fType; //文件类型，0 为文件夹，1 为文件
    char content[MAX_CONTENT_]; //文件内容
    int fatherID; //父文件 ID
    int children[MAX_FILE_PER_LAYER]; //子文件序列，包括所有子文件 ID
    int childrenNumber; //子文件数目
};
struct fBlock blocks[MAX_FILE_NUM];
int IDLog[MAX_FILE_NUM]; //文件位图，0 表示该位置为空
int currentFileID = 0; //当前所处文件/文件夹 ID
int fileNum = 0; //当前文件数目，实现对文件系统的实时监控
```

3.6.4 功能函数及简介

1. void FsShowCmd(): 用于展示文件系统的交互页面
2. void InitBlock(): 初始化一个空的文件块，该函数会在创建新文件时调用
3. void toStr3(char* temp, int i): 将整型数 i 转化为 3 位字符串 temp，如果 i 小于三位会自动

补 0, i 大于三位会只转化后三位。该函数是为了将文件块的 ID 以统一的格式记录到系统文件内。

4. int toInt(char* temp): 将 temp 转化为整数, 只支持 3 位的转化。

5. void WriteDisk(int len): 写硬盘函数, len 为待写入的字符串长度。该函数首先将 blocks[] 中的信息转化为一个长字符串, 然后将该字符串写进系统文件。该函数在 \$ sv 指令调用。

6. int ReadDisk(): 读硬盘函数。该函数会将系统文件中的长字符串依次转化为正确类型并赋值到 blocks[] 中。该函数会在文件系统初始化前调用。

7. void FSinit(): 文件系统初始化函数。该函数将初始化内存中的所有文件块, 并且创建一个名为 “home” 的根文件夹, 所有操作均在该文件夹目录下进行。

8. int CreateFile(char* fileName, int fileType): 创建文件函数, 返回 1 表示创建成功, 0 表示创建失败, 该函数会在 \$ touch 指令后得到调用。

9. void showFileList(): 以列表形式展示当前文件夹下的所有文件, 该函数会在 \$ ls 指令后得到调用。

10. int SearchFile(char* name): 根据文件名查找文件 ID, 如果找到则返回文件 ID, 未找到则返回错误标识-2;

11. void DeleteFile(int ID): 删除指定 ID 的文件。该函数中, 被删除的文件的所有子文件也会递归调用该函数进行删除, 同时该函数还会对待删除文件的父文件中的相应项进行删除。该函数会在 \$ rm 指令中调用

12. void filemanagerprocess(int fd_stdin): 文件系统的主函数, 该函数会调用相应功能函数完成初始化, 然后进入 while(1) 循环等待用户输入, 根据不同输入调用不同的函数, 实现文件系统的正常工作。

3.6.5 截屏示例

主界面



展示当前目录下文件（图中文件为历史文件保存）：

```

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

#
# $ touch [name]      : Create a file      #
# $ mkdir [name]      : Create a folder    #
# $ cd [name]         : Enter sub path     #
# $ bk               : Return superior path #
# $ ls               : Show file list      #
# $ rm [name]        : Remove file        #
# $ sv               : Save to disk        #
# $ clear            : Clear screen        #
# $ help            : Show commands       #
# $ quit            : quit FILE MANAGER   #
#
#                               #
#       Supported by DOGE R&D TEAM       #
#       ALL RIGHT REVERSED              #
#                               #
=====

home $ ls
[home] File List:

=====
      NAME : TYPE : ID
  README : .TXT : 1
     LYF : .FOLDER : 2
=====

home $

```

IPS: 85,694M A: NUM CAPS SCRL HD:0-M

创建文件:

```

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

#
#       Supported by DOGE R&D TEAM       #
#       ALL RIGHT REVERSED              #
#                               #
=====

home $ ls
[home] File List:

=====
      NAME : TYPE : ID
  README : .TXT : 1
     LYF : .FOLDER : 2
=====

home $ touch test
File [test] created successfully.
home $ ls
[home] File List:

=====
      NAME : TYPE : ID
  README : .TXT : 1
     LYF : .FOLDER : 2
     test : .TXT : 4
=====

home $

```

IPS: 87,572M A: NUM CAPS SCRL HD:0-M

文件读写:

```
Bochs x86 emulator, http://bochs.sourceforge.net/
=====
NAME | TYPE | ID
-----
README | .TXT | 1
LYF | .FOLDER | 2
=====

home $ touch test
File [test] created successfully.
home $ ls
[home] File List:
=====
NAME | TYPE | ID
-----
README | .TXT | 1
LYF | .FOLDER | 2
test | .TXT | 4
=====

home $ cd test
You have opened a TXT file
You can use the following commands to manipulate the file:
$ u --- update the file content
$ c --- check the file content
$ q --- close the file
home/test $
```

```
Bochs x86 emulator, http://bochs.sourceforge.net/
=====
NAME | TYPE | ID
-----
README | .TXT | 1
LYF | .FOLDER | 2
test | .TXT | 4
=====

home/test $ u
Please enter the updated content:
(The maximum number of characters supported by DOGE is 50)
WELCOME
Update complete.
You have opened a TXT file
You can use the following commands to manipulate the file:
$ u --- update the file content
$ c --- check the file content
$ q --- close the file
home/test $ c
=====DOGE=====
WELCOME
=====DOGE=====

You have opened a TXT file
You can use the following commands to manipulate the file:
$ u --- update the file content
$ c --- check the file content
$ q --- close the file
home/test $
```

创建目录:

```

Bochs x86 emulator, http://bochs.sourceforge.net/
home/test $ c

=====DOGE=====
WELCOME
=====DOGE=====

You have opened a TXT file
You can use the following commands to manipulate the file:
$ u --- update the file content
$ c --- check the file content
$ q --- close the file
home/test $ q
home $ mkdir wow
Folder [wow] created successfully.
home $ ls
[home] File List:

=====
NAME | TYPE | ID
-----
README | .TXT | 1
LYF | .FOLDER | 2
test | .TXT | 4
wow | .FOLDER | 5
=====

帮助
home $
IPS: 79,120M | A: NUM | CAPS | SCRL | HD:0-M |

```

进入目录和退出目录:

```

Bochs x86 emulator, http://bochs.sourceforge.net/
=====DOGE=====
WELCOME
=====DOGE=====

You have opened a TXT file
You can use the following commands to manipulate the file:
$ u --- update the file content
$ c --- check the file content
$ q --- close the file
home/test $ q
home $ mkdir wow
Folder [wow] created successfully.
home $ ls
[home] File List:

=====
NAME | TYPE | ID
-----
README | .TXT | 1
LYF | .FOLDER | 2
test | .TXT | 4
wow | .FOLDER | 5
=====

home $ cd wow
home/wow $ bk
home $
IPS: 87,816M | A: NUM | CAPS | SCRL | HD:0-M |

```

删除文件:


```
Bochs x86 emulator, http://bochs.sourceforge.net/
A: B: CD
Folder [wow] created successfully.
home $ ls
[home] File List:
=====
NAME | TYPE | ID
-----
README | .TXT | 1
LYF | .FOLDER | 2
test | .TXT | 4
wow | .FOLDER | 5
=====
home $ cd wow
home/wow $ bk
home $ rm test
The file or folder has been successfully removed.
home $ ls
[home] File List:
=====
NAME | TYPE | ID
-----
README | .TXT | 1
LYF | .FOLDER | 2
wow | .FOLDER | 5
=====
home $
TPS: 87.064M | A: NUM | CAPS | SCRI | HT:0-M |
```

显示帮助:

```
Bochs x86 emulator, http://bochs.sourceforge.net/
A: B: CD
home $ help
=====
# DOGE #
# FILE MANAGER #
# [COMMAND] [FUNCTION] #
# $ touch [name] | Create a file #
# $ mkdir [name] | Create a folder #
# $ cd [name] | Enter sub path #
# $ bk | Return superior path #
# $ ls | Show file list #
# $ rm [name] | Remove file #
# $ sv | Save to disk #
# $ clear | Clear screen #
# $ help | Show commands #
# $ quit | quit FILE MANAGER #
# Supported by DOGE R&D TEAM #
# ALL RIGHT REVERSED #
=====
home $
TPS: 61.928M | A: NUM | CAPS | SCRI | HT:0-M |
```

3.7. 进程管理系统

3.7.1 基本功能

(1).在控制台命令行输入 pm 进入进程管理系统。

- (2). 在进程系统控制台输入 `ps`，输出所有进程相关信息，包括进程 ID，进程名称，进程的优先级以及运行的状态。
- (3). 命令行输入 `kill`，再输入结束进程的 `id`。ID 合法则进程可以被结束。如果进程删除了系统级进程，处于安全考虑会被禁止。如果进程删除 `TestA` 进程，将会导致进程系统关闭，因此防止中断，也会被禁止删除
- (4). 命令行输入 `create`，再输入启动进程的 `id`。如果 `id` 合法且未在运行，则进程被启动。
- (5). 命令行输入 `help`，将展示进程管理系统帮助界面

3.7.2 函数实现

- 1.`void createprocess(int id);`负责判断当前 `id` 对应的进程是否在运行，如果处于关闭状态的进程，则修改进程的优先级，将其修改成大于 0 的值
- 2.`void killprocess(int id);` 负责判断当前 `id` 对应的进程是否在运行，如果处于运行状态且是可以被关闭的用户级的进程，则修改进程的优先级，将其修改为 0。
- 3.`void processinfo();`根据系统进程表 `proc_table` 记录的所有进程依次输出进程的 `id`,进程名，进程的优先级以及进程是否在运行
- 4.`void showcommands();`在屏幕中打印出进程管理系统所有操作指令及其解释，方便用户使用本系统

3.7.3 截屏示例

主界面及 `help` 指令界面

```

Bochs x86-64 emulator, http://bochs.sourceforge.net/
=====
#                                     #
#                                     #
#               DOGE                  #
#                                     #
#               PROCESS MANAGER       #
#                                     #
#                                     #
#               [COMMAND]             #
#                                     #
#               [FUNCTION]            #
#                                     #
#               $ ps                  #
#               $ kill                #
#               $ create              #
#               $ help                #
#               $ clear               #
#               $ exit                #
#                                     #
#                                     #
#               Powered by chuckle    #
#               ALL RIGHT REVERSED    #
#                                     #
=====
DOGE/PROCESS MANAGER $
IPS: 56.571M  A: NUM CAPS SCRL HD:0-M

```

`create` 指令

```

DOGE/PROCESS MANAGER $ create
please input process id:5
The Process begins to running.
DOGE/PROCESS MANAGER $ ps
=====
ProcessID | ProcessName | Priority | Running?
-----
0 | TTY | 15 | YES
1 | SYS | 15 | YES
2 | HD | 15 | YES
3 | FS | 15 | YES
4 | TestA | 5 | YES
5 | TestB | 5 | YES
6 | TestC | 5 | YES
=====
DOGE/PROCESS MANAGER $

```

IPS: 56.792M A: NUM CAPS SCRL HD:0-M

kill 指令

```

DOGE/PROCESS MANAGER $ kill
please input process id:4
The Process is our main process.You can not kill it.
DOGE/PROCESS MANAGER $ kill
please input process id:5
The Process is killed.
DOGE/PROCESS MANAGER $ ps
=====
ProcessID | ProcessName | Priority | Running?
-----
0 | TTY | 15 | YES
1 | SYS | 15 | YES
2 | HD | 15 | YES
3 | FS | 15 | YES
4 | TestA | 5 | YES
5 | TestB | 0 | NO
6 | TestC | 5 | YES
=====
DOGE/PROCESS MANAGER $

```

IPS: 59.468M A: NUM CAPS SCRL HD:0-M

ps 指令:

```

DOGE/PROCESS MANAGER $ ps
=====
ProcessID | ProcessName | Priority | Running?
-----
0 | TTY | 15 | YES
1 | SYS | 15 | YES
2 | HD | 15 | YES
3 | FS | 15 | YES
4 | TestA | 5 | YES
5 | TestB | 5 | YES
6 | TestC | 5 | YES
=====

```

4. 成员分工

成员	分工
李源峰 1852448	Orange's 源码研读、文件系统设计及调试、定时器 1 设计及调试、小组成员成果整合、文档设计和整合
吕梓源 1851489	Orange's 源码研读、内存管理内容研究、进程管理系统设计及调试
周子怡 1854139	计算器设计及调试、扫雷设计及调试、五子棋设计及调试、答辩 PPT 整合

何娴 1852342	日历设计及调试、猜数字设计及调试、推箱子设计及调试
杜玫 1853316	计时器 2 设计及调试、2048 游戏设计及调试、石头剪刀布设计及调试

5. 项目开发历程

7.20-8.3: Orange' s 源码研读, 基础知识学习, 环境配置, 试运行样例代码并修复 Bug。

8.3: 分配任务。

8.3~8.10: 文件系统、进程管理系统开始开发, 用户级应用开始在 linux 平台上开发。

(8.8: 文件系统更改开发模式, 转为存储一个系统文件的多级系统。)

8.10-8.17: 用户级应用开始向 DOGE 移植, 修复各种 bug.

8~17-8.19: 成果汇总, 修复 bug, 完成 DOGE 开发

(8.19: DOGE 开发完成)

8.19-8.21: 细节修改, 文档撰写及汇总, ppt 制作及汇总