
操作系统课程设计文档

LightningOS 一个基于 OrangeS 的操作系统实现

姓名：郑柯凡

学号：1950072

指导教师：王冬青

专业：软件工程

同济大学

Tongji University

目录

1 项目概述	3
1.1 项目简介	3
1.2 项目配置	3
2 功能实现	4
2.1 简单用户级应用	4
2.1.1 开机动画	4
2.1.1.1 界面展示	4
2.1.1.2 核心代码	4
2.1.2 命令提示	6
2.1.2.1 界面展示	6
2.1.2.2 核心代码	6
2.1.3 进制转换	8
2.1.3.1 界面展示	8
2.1.3.2 核心代码	9
2.1.4 五子棋	11
2.1.4.1 界面展示	11
2.1.4.2 核心代码	13
2.1.5 国际跳棋	15
2.1.5.1 界面展示	15
2.1.5.2 核心代码	17
2.2 复杂用户级应用	19
2.2.1 计算器	19
2.2.1.1 界面展示	19
2.2.1.2 核心代码	21
2.2.2 贪吃蛇	23
2.2.2.1 界面展示	23
2.2.2.2 核心代码	25
2.2.3 游戏 2048	26
2.2.3.1 界面展示	26
2.2.3.2 核心代码	28
2.3 系统级应用	30
2.3.1 进程管理	30
2.3.1.1 界面展示	30
2.3.1.2 核心代码	30
2.3.2 文件系统	31
2.3.2.1 界面展示	31
2.3.2.2 代码实现	37
3 模块修改	41
3.1 进程调度模块	41
3.2 控制台模块	42
3.3 输入输出系统	44

1 项目概述

1.1 项目简介

LightningOS 项目参考《Orange' S 一个操作系统的实现》，在 Orange' S 操作系统的基础上实现一个功能更加丰富，交互更加友好的操作系统。本项目添加了一些基础的操作系统功能。如开机启动动画，命令行提示等。同时，本项目中实现了 6 个用户级应用，分别是：进制转换应用、计算器、国际跳棋、五子棋、贪吃蛇以及 2048。

在原操作系统的基础上，本项目对控制台、进程模块和文件系统模块进行修改和完善。该操作系统实现了三个控制台，独立运行各自的进程，互不干扰。进程模块增加了进程查看功能，可以查看存在进程的进程号、优先级、创建用户等信息，并对进程调度算法进行优化。文件系统模块将原本的扁平文件系统增加目录深度，并实现基本的文件增删改查功能。

1.2 项目配置

编写语言：汇编、C

开发环境：Ubuntu 18.04.5 LTS (Windows 下的 Vmware 虚拟机中)

开发工具：VS Code、Bochs x86 Emulator 2.6

运行环境：Bochs x86 Emulator 2.6

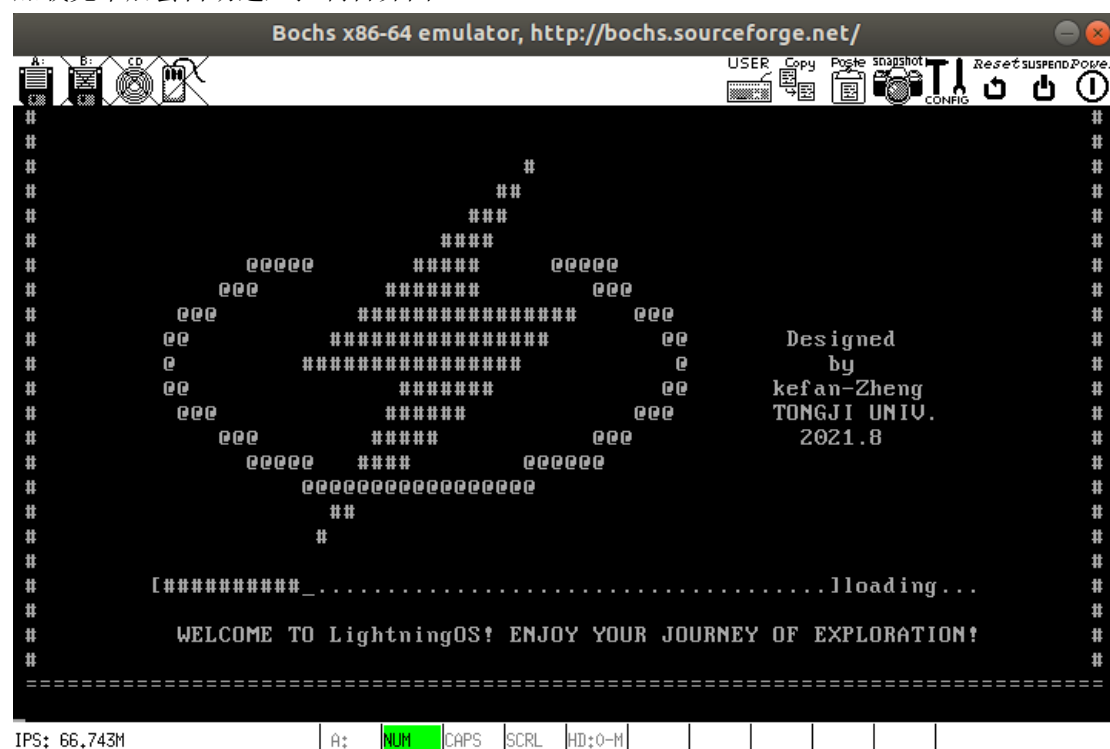
2 功能实现

2.1 简单用户级应用

2.1.1 开机动画

2.1.1.1 界面展示

开机界面，操作系统会播放动画并展示操作系统标识。进度条表示操作系统的加载进度，加载完毕后会自动进入控制台界面。



2.1.1.2 核心代码

```
1.  /*+++++Kefan-Zheng, 2021
2.
3.  +++++*/
4.  #include "type.h"
5.  #include "stdio.h"
6.  #include "const.h"
7.  #include "protect.h"
8.  #include "string.h"
9.  #include "fs.h"
10. #include "proc.h"
11. #include "tty.h"
```

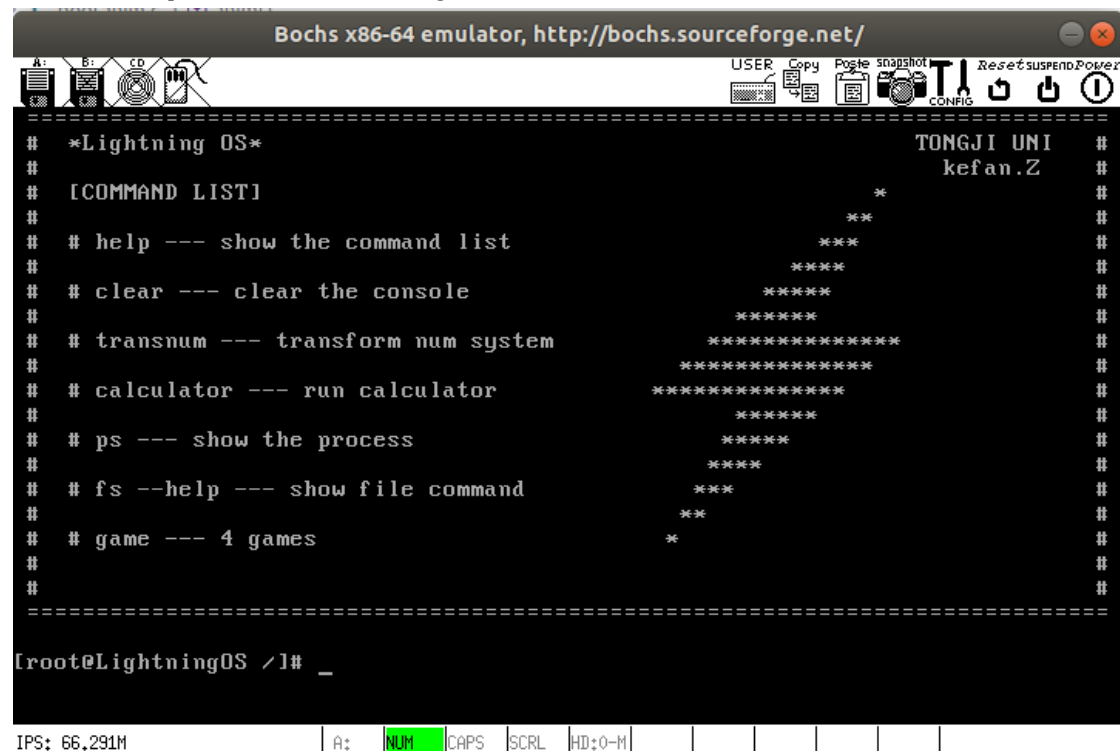
```
12. #include "console.h"
13. #include "global.h"
14. #include "proto.h"
15.
16. #define DELAY_TIME 2500
17.
18. void anim(){
19.     char str[15] = "loading...";
20.     clear();
21.     printf(" =====\n");
22.     printf(" #                                     #\n");
23.     printf(" #                                     #\n");
24.     printf(" #                                     #\n");
25.     printf(" #                                     ##\n");
26.     printf(" #                                     ###\n");
27.     printf(" #                                     ####\n");
28.     printf(" #          @@@@          #####          @@@@          #\n");
29.     printf(" #          @@@          #####          @@@          #\n");
30.     printf(" #          @@          #####          @@          #\n");
31.     printf(" #          @          #####          @          Designed\n");
32.     printf(" #          @          #####          @          by\n");
33.     printf(" #          @@          #####          @@          kefan-Zheng\n");
34.     printf(" #          @@@          #####          @@@          TONGJI UNIV.\n");
35.     printf(" #          @@@          #####          @@@          2021.8\n");
36.     printf(" #          @@@@          #####          @@@@@@          #\n");
37.     printf(" #          @@@@@@@@@@@@@@@@@@@@@@          #\n");
38.     printf(" #          ##          #\n");
39.     printf(" #          #          #\n");
40.     printf(" #          #\n");
41.     printf(" #          [/.]loading... \n");
42.     printf(" #          \n");
43.     printf(" #          WELCOME TO LightningOS! ENJOY YOUR JOURNEY OF EXPLORATION! \n");
44.     printf(" #          \n");
45.     printf(" =====\n");
46.     milli_delay(DELAY_TIME);
47.     clear();
48.     .....
```

先使用 `clear()` 函数将控制台屏幕对应的显存清零。再用 `printf()` 函数将开机动画的一帧打印在控制台中。最后再用 `milli_delay()` 函数，其通过一个获取时钟周期数的系统调用延迟程序的运行。通过连续逐帧打印的方法实现动画的播放。

2.1.2 命令提示

2.1.2.1 界面展示

命令提示界面打印出一系列有意义的命令语句，通过[命令 --- 作用]的形式告诉用户操作系统支持的命令以及其作用。其中，包括帮助命令[help]、清除控制台命令[clear]、进制转换命令[transnum]、计算器命令[calculator]、查看进程命令[ps]、文件系统帮助命令[fs --help]以及4个游戏命令[game]。



2.1.2.2 核心代码

```
1. void TestA()
2. {
3.     //文件部分变量
4.     char curdir[512] = "/";
5.     char rdbuf[256];
6.
7.     //char command_3[100], command_4[100], command_5[100];
8.     int fd_stdin = open("/dev/tty0", O_RDWR);
9.     assert(fd_stdin == 0);
10.    int fd_stdout = open("/dev/tty0", O_RDWR);
11.    assert(fd_stdout == 1);
12.    anim();
13.    commandList();
14.}
```

```
15. while (1) {
16.     printf("[%s@LightningOS %s]# ",sys_user, curdir);
17.     boundary = tty_table[0].console->cursor;
18.     int r = read(fd_stdin, rdbuf, 256);
19.     rdbuf[r] = 0;
20.     if(parsecmd(rdbuf, curdir)==0)
21.     {
22.         if (!strcmp(rdbuf, "help"))
23.         {
24.             clear();
25.             commandList();
26.         }
27.         else if (!strcmp(rdbuf, "clear"))
28.         {
29.             clear();
30.         }
31.         else if (!strcmp(rdbuf, "game"))
32.         {
33.             gameList();
34.         }
35.         else if (!strcmp(rdbuf, "calculator"))
36.         {
37.             clear();
38.             calculator_main(&fd_stdin);
39.             commandList();
40.             continue;
41.         }
42.         else if (!strcmp(rdbuf, "ps"))
43.         {
44.             processManage();
45.         }
46.         else if (!strcmp(rdbuf, "fs --help"))
47.         {
48.             filecmdList();
49.         }
50.         else if (!strcmp(rdbuf,"draughts"))
51.         {
52.             draughts_main(&fd_stdin);
53.             clear();
54.         }
55.         else if (!strcmp(rdbuf,"2048"))
56.         {
57.             g2048_main(&fd_stdin);
58.             clear();
```

```
59.     }
60.     else if (!strcmp(rdbuff,"snake"))
61.     {
62.         snake_main(&fd_stdin);
63.         clear();
64.     }
65.     else if (!strcmp(rdbuff,"transnum"))
66.     {
67.         numsys_main(&fd_stdin);
68.     }
69.     else if (!strcmp(rdbuff,"gomoku"))
70.     {
71.         gomoku_main(&fd_stdin);
72.         clear();
73.     }
74.     else    //找不到所输入的指令
75.     {
76.         noSuchCom();
77.     }
78. }
79. }
80. for(ever);
81. }
```

先打印命令提示界面，再调用 `read()` 系统调用不断读取用户的输入，并做出判断调用相应的功能函数。

2.1.3 进制转换

2.1.3.1 界面展示

输入进制转换的命令[transnum]进入进制转换模式。系统将会提示输入数字的目标进制。待输入完毕后，系统将会要求用户输入准备转换的数字以及其当前的进制。当用户全部输入完毕后，系统就会自己完成计算并输出进制转换之后的数字。

此处将十进制的数字 8 转换成二进制编码，答案正确。本系统支持所有二十进制以内的进制转换。

```
[root@LightningOS ~]# transnum
Please input the target system:2
Please input the num and its sys:8 10
ans:1000
Do u want to try again?(y/n):y
```


2.1.3.2 核心代码

```
1. while(numsysend == 0)
2. {
3.     int sys, num, orsys;
4.     int len = 0, single;
5.     int i = 0;
6.     char res[100];
7.     char rdbuff[128];
8.     int rdbufflen;
9.
10.    printf("\n");
11.    printf("Please input the target system:");
12.    rdbufflen = read(*fd_stdin, rdbuff, 128);
13.    rdbuff[rdbufflen] = '\0';
14.    sys = atoi(rdbuff);
15.    printf("Please input the num and its sys:");
16.    rdbufflen = read(*fd_stdin, rdbuff, 128);
17.    char tmpary[5][64];
18.    int first = 0;
19.    int second = 0;
20.    for(int j = 0; j < rdbufflen; j++)
21.    {
22.        if(rdbuff[j] == ' ')
23.        {
24.            tmpary[first][second] = '\0';
25.            first = 1;
26.            second = 0;
27.            j++;
28.        }
29.        tmpary[first][second] = rdbuff[j];
30.        second++;
31.    }
32.    tmpary[first][second] = '\0';
33.    num = atoi(tmpary[0]);
34.    orsys = atoi(tmpary[1]);
35.    // 将输入的数字转换成10进制的
36.    int tmp = 0;
37.    for (len; len <= 50; len++)
38.    {
39.        single = num % 10;
40.        tmp = tmp + single * pow(orsys, len);
41.        num = num / 10;
42.    }
```

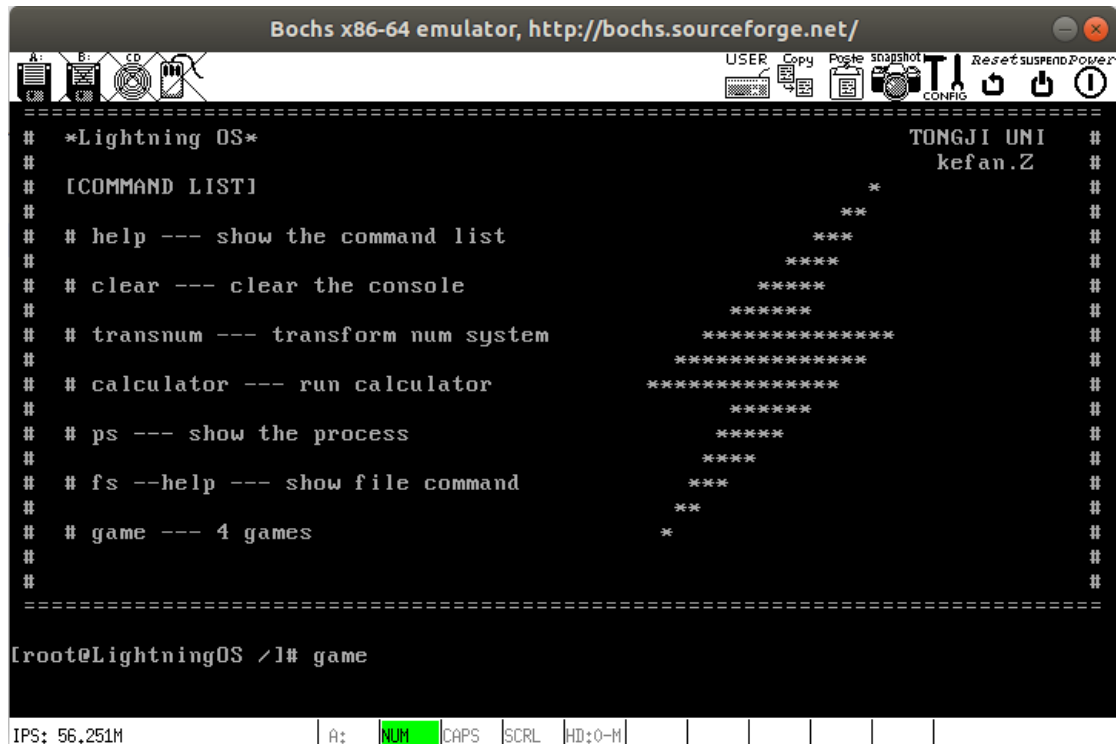
```
43. char p[30] = { 0,1,2,3,4,5,6,7,8,9,'A','B','C','D','E','F','G','H','I','J' };
44. if (sys <= 10)
45. {
46.     while (tmp >= sys)
47.     {
48.         res[i] = tmp % sys;
49.         tmp = tmp / sys;
50.         i = i + 1;
51.     }
52.     res[i] = tmp;
53.     printf("ans:");
54.     for (i; i >= 0; i--)
55.     {
56.         printf("%d", res[i]);
57.     }
58. }
59. else if (10 < sys && sys <= 20)
60. {
61.     while (tmp >= sys)
62.     {
63.         res[i] = p[tmp % sys];
64.         tmp = tmp / sys;
65.         i = i + 1;
66.     }
67.     res[i] = p[tmp];
68.     printf("ans:");
69.     for (i; i >= 0; i--)
70.     {
71.         if (res[i] < 60)
72.         {
73.             printf("%d", res[i]);
74.         }
75.         else
76.         {
77.             printf("%c", res[i]);
78.         }
79.     }
80. }
81. printf("\nDo u want to try again?(y/n):");
82. read(*fd_stdin,rdbuff,128);
83. if(rdbuff[0] == 'n')
84. {
85.     numsysend = 1;
86. }
```

```
87. }
```

2.1.4 五子棋

2.1.4.1 界面展示

控制台输入[game]命令，控制台将会给出游戏列表。



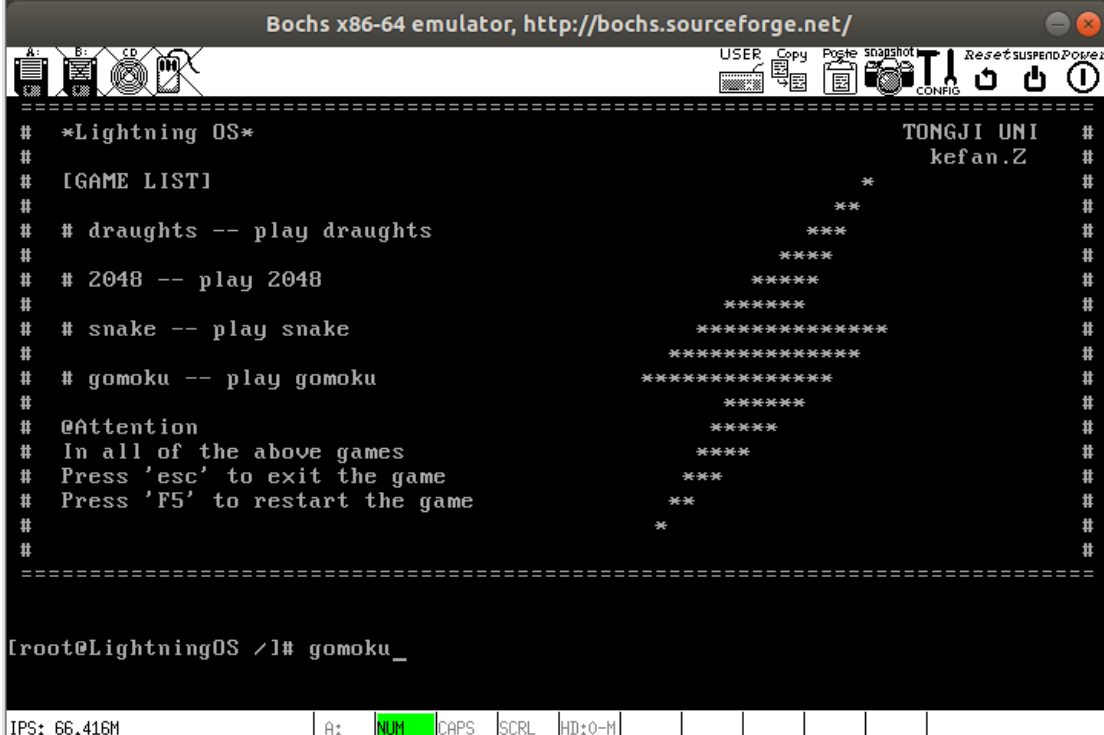
```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

=====
# *Lightning OS*                                TONGJI UNI #
# [COMMAND LIST]                                kefan.Z  #
# # help --- show the command list              *         #
# # clear --- clear the console                 **        #
# # transnum --- transform num system            ***       #
# # calculator --- run calculator                ****      #
# # ps --- show the process                     *****    #
# # fs --help --- show file command              *         #
# # game --- 4 games                            *         #
# #                                              *         #
=====

[root@LightningOS /]# game
```

IPS: 56.251M A: NUM CAPS SCRL HD: 0-M

游戏列表界面给出 4 个对应的游戏以及启动游戏的命令。同时，在列表下方也给出了 4 个游戏通用的一些操作，比如说按 f5 键重置游戏，按 esc 键退出游戏。



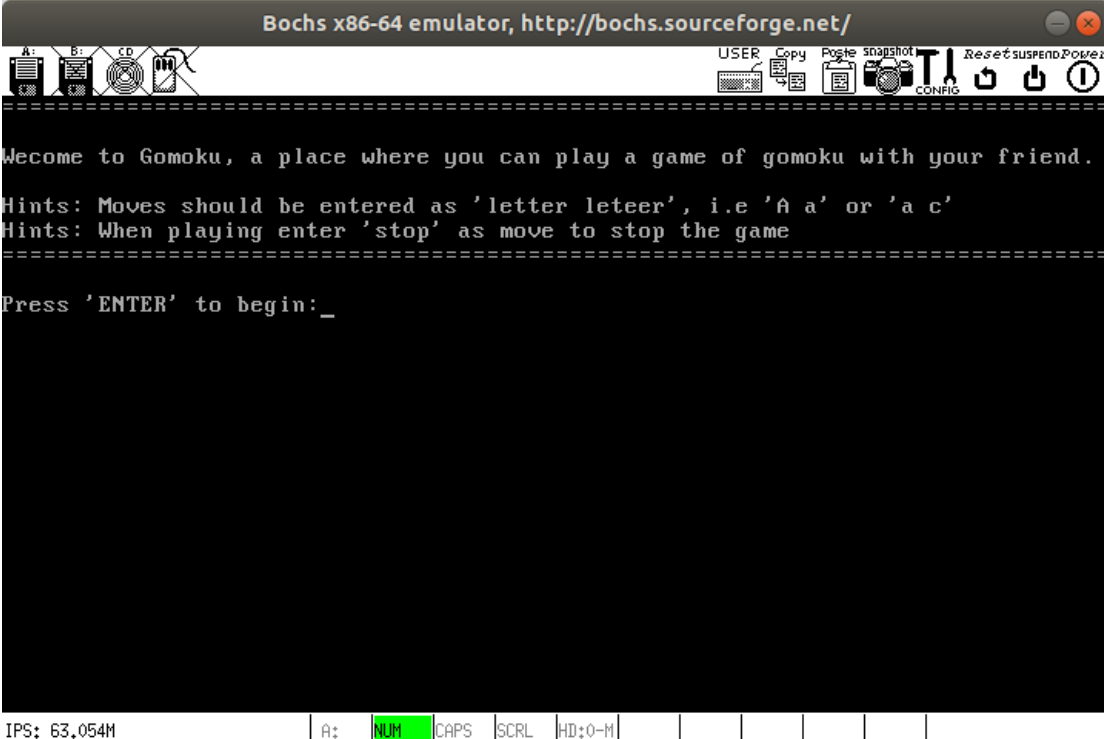
```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

=====
#  *Lightning OS*                                TONGJI UNI #
#                                                    kefan.Z  #
# [GAME LIST]                                     *          #
#                                                    **         #
# # draughts -- play draughts                    ***        #
#                                                    ****       #
# # 2048 -- play 2048                             *****    #
#                                                    *          #
# # snake -- play snake                           ****         #
#                                                    *          #
# # gomoku -- play gomoku                         ****         #
#                                                    *          #
# @Attention                                     ****         #
# In all of the above games                       ****         #
# Press 'esc' to exit the game                     ***          #
# Press 'F5' to restart the game                   **           #
#                                                    *            #
#                                                    #
=====

[root@LightningOS /]# gomoku_

=====
IPS: 66.416M  A: NUM CAPS SCRL HD:0-M
```

输入五子棋游戏对应的命令[gomoku]后，先进入玩家欢迎界面，此界面介绍了五子棋游戏的操作规则：输入坐标[x1 y1]放置棋子，输入[stop]退出游戏等。并等待玩家按下 ENTER 键开始游戏。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

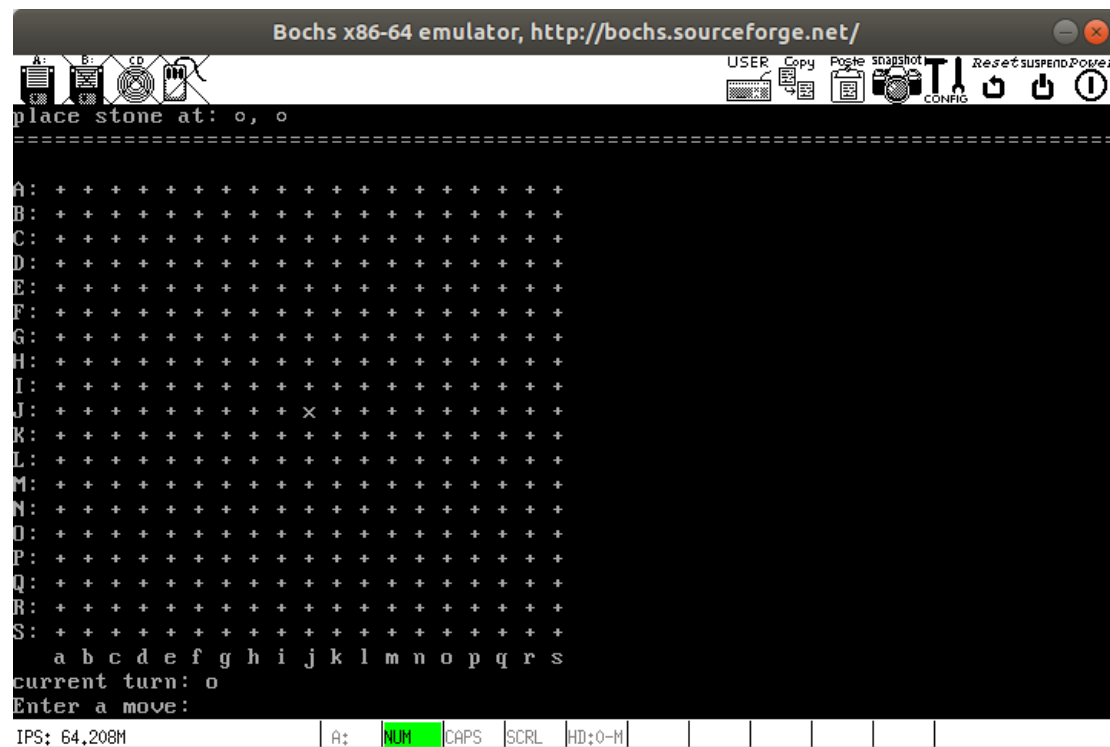
=====
Welcome to Gomoku, a place where you can play a game of gomoku with your friend.

Hints: Moves should be entered as 'letter leteer', i.e 'A a' or 'a c'
Hints: When playing enter 'stop' as move to stop the game
=====

Press 'ENTER' to begin:_

=====
IPS: 63.054M  A: NUM CAPS SCRL HD:0-M
```

玩家按下 ENTER 键后, 游戏正式开启。玩家只需按照棋盘输入对应坐标即可。值得注意的是, 这是一个双人游戏, 黑棋白棋均需用户输入。



2.1.4.2 核心代码

```
1. clear();
2.     printf("=====\n");
3.     printf("Wecome to Gomoku, a place where you can play a game of gomoku with your fr
   iend. \n");
4.     printf("Hints: Moves should be entered as 'letter leteer', i.e 'A a' or 'a c'\n");
5.     printf("Hints: When playing enter 'stop' as move to stop the game\n");
6.     printf("=====\n");
7.     printf("Press 'ENTER' to begin:");
8.     while(ifgomokustart == 0){}
9.     printf("OK! Game starts.\n");
10.    while (!is_over)
11.    {
12.        if(ifgomokustart = 0){ifgomokustart = 1;}
13.        clear();
14.        memset(board, '0', sizeof(board));
15.        is_won = 0;
16.        player = 'x';
17.        while (!is_won)
18.        {
```

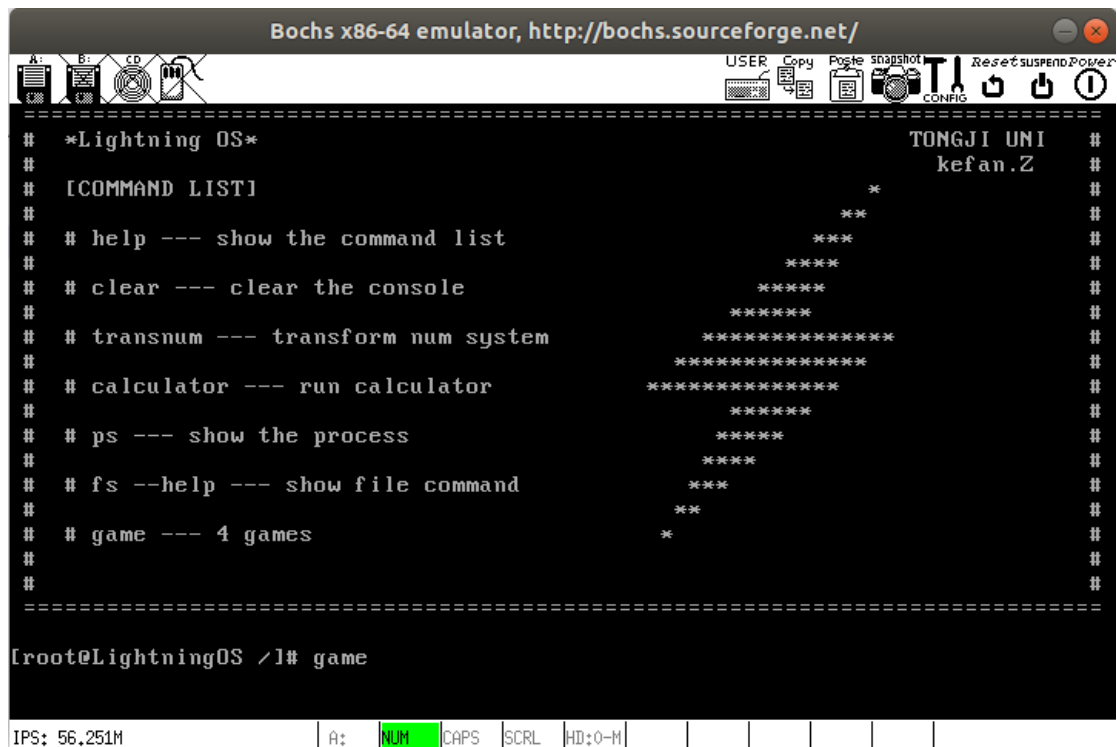
```
19.         printf("=====\n");
20.         draw_board(board);
21.         printf("current turn: %c\n", player);
22.         char move[10];
23.
24.         printf("Enter a move: ");
25.         int r = read(*fd_stdin, move, 128);
26.         move[r] = 0;
27.         if(!strcmp(move,"exit")){ gomexit();break; }
28.         else if(!strcmp(move,"restart")){ gomrestart(); continue;}
29.         char move_x = move[0];
30.         char move_y = move[2];
31.         if (!isalpha(move_x) || !isalpha(move_y))
32.         {
33.             printf("Invalid move, must enter letters.");
34.         }
35.         int x;
36.         int y;
37.         x = toupper(move_x) - 65;
38.         y = toupper(move_y) - 65;
39.         printf("place stone at: %c, %c \n", x, y);
40.         if (make_move(x, y))
41.         {
42.             result = is_winning();
43.             if (result != '0')
44.             {
45.                 is_won = 1;
46.             }
47.             else
48.             {
49.                 player = (player == 'x') ? 'o' : 'x';
50.             }
51.         }
52.         // end turn clean up
53.     }
54.     if (!is_over)
55.     {
56.         (result == '1') ? printf("Game tied, starting a new game...\n") :
57.         ((result == 'x') ? printf("x won the game, starting a new game...\n") :
58.         printf("o won the game, starting a new game...\n"));
59.         printf("Press esc/f5 to exit/restart.\n");
60.         while(ifgomokustart == 1){}
61.     }
```

```
62.         printf("\n\n      ");
63.     }
```

2.1.5 国际跳棋

2.1.5.1 界面展示

控制台输入[game]命令，控制台将会给出游戏列表。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

=====
# *Lightning OS*                                TONGJI UNI #
#                                                    kefan.Z  #
# [COMMAND LIST]                                     *          #
#                                                    ***        #
# # help --- show the command list                 ****       #
#                                                    *****     #
# # clear --- clear the console                    ****        #
# # transnum --- transform num system              ****        #
#                                                    ****        #
# # calculator --- run calculator                  ****        #
#                                                    ****        #
# # ps --- show the process                       ****        #
#                                                    ****        #
# # fs --help --- show file command                ****        #
#                                                    ****        #
# # game --- 4 games                               *          #
#                                                    #          #
# =====
[root@LightningOS /]# game
```

IPS: 56.251M | A: NUM | CAPS | SCRL | HD: 0-M |

游戏列表界面给出 4 个对应的游戏以及启动游戏的命令。同时，在列表下方也给出了 4 个游戏通用的一些操作，比如说按 f5 键重置游戏，按 esc 键退出游戏。



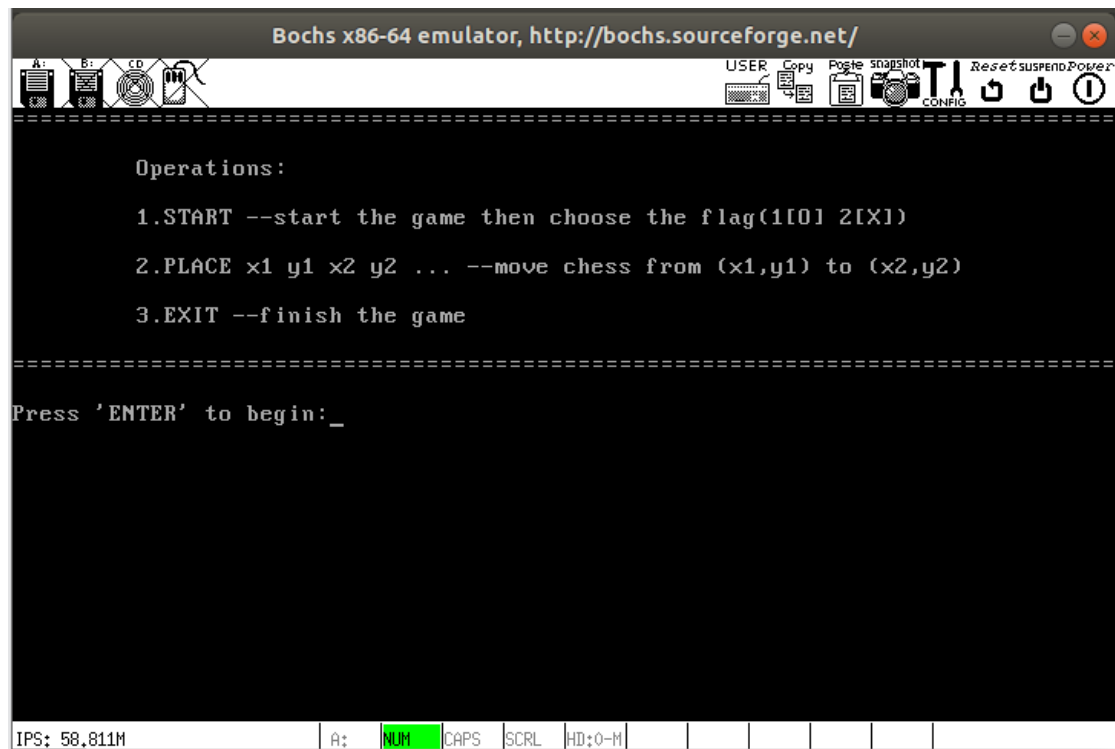
```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

=====
# *Lightning OS*                                TONGJI UNI #
#                                                    kefan.Z  #
# [GAME LIST]                                     *          #
#                                                    ***         #
# # draughts -- play draughts                    *****      #
#                                                    ****          #
# # 2048 -- play 2048                             ****         #
#                                                    ****          #
# # snake -- play snake                          ****          #
#                                                    ****          #
# # gomoku -- play gomoku                       ****          #
#                                                    ****          #
# @Attention                                     ****          #
# In all of the above games                      ****          #
# Press 'esc' to exit the game                   ****          #
# Press 'F5' to restart the game                 **          #
#                                                    *            #
#                                                    #
=====

[root@LightningOS /]# draughts_

IPS: 58,358M  A: NUM CAPS SCRL HD:0-M
```

输入国际跳棋的启动命令[draughts]后，同样进入游戏欢迎界面。在该界面，系统介绍了该游戏的几种基本操作：[START]启动游戏并选择黑棋或白棋、[PLACE]操作用于移动棋子。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

=====
Operations:

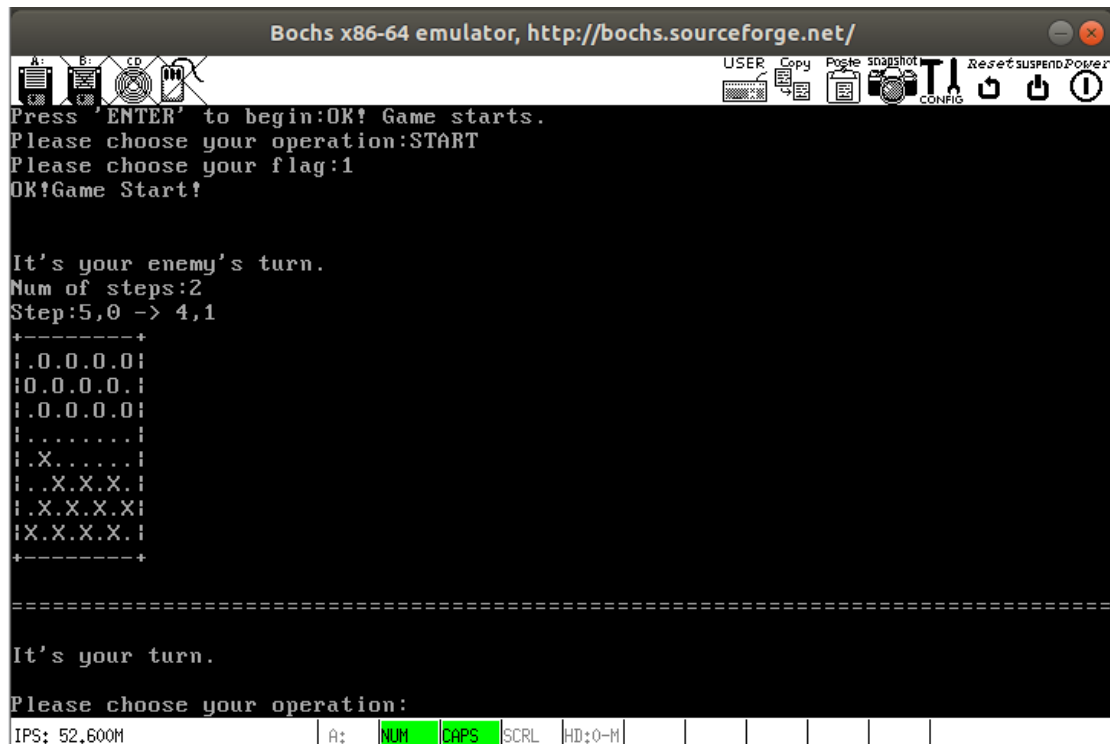
1.START --start the game then choose the flag(1[O] 2[X])
2.PLACE x1 y1 x2 y2 ... --move chess from (x1,y1) to (x2,y2)
3.EXIT --finish the game

=====

Press 'ENTER' to begin:_

IPS: 58,811M  A: NUM CAPS SCRL HD:0-M
```


值得注意的是，该游戏是一个单人游戏。玩家将会对战一个国际跳棋 AI，因此玩家只需在自己的回合选择操作即可。同时，系统也会给出对局信息如：此时是谁的回合。



2.1.5.2 核心代码

```
1. void loop(int *fd_stdin)
2. {
3.     char tag[10] = { 0 };
4.     struct Command command =
5.     {
6.         .x = {0},
7.         .y = {0},
8.         .numStep = 0
9.     };
10.    int status;
11.    while (TRUE)
12.    {
13.        memset(tag, 0, sizeof(tag));
14.        printf("Press esc/f5 to exit/restart\n");
15.        printf("Please choose your operation:");
16.        read(*fd_stdin, tag, sizeof(tag));
17.        if (strcmp(tag, START) == 0)
18.        {
19.            printf("Please choose your flag:");
20.            char myChooseFlag[2];
21.            read(*fd_stdin, myChooseFlag, sizeof(myChooseFlag));
```

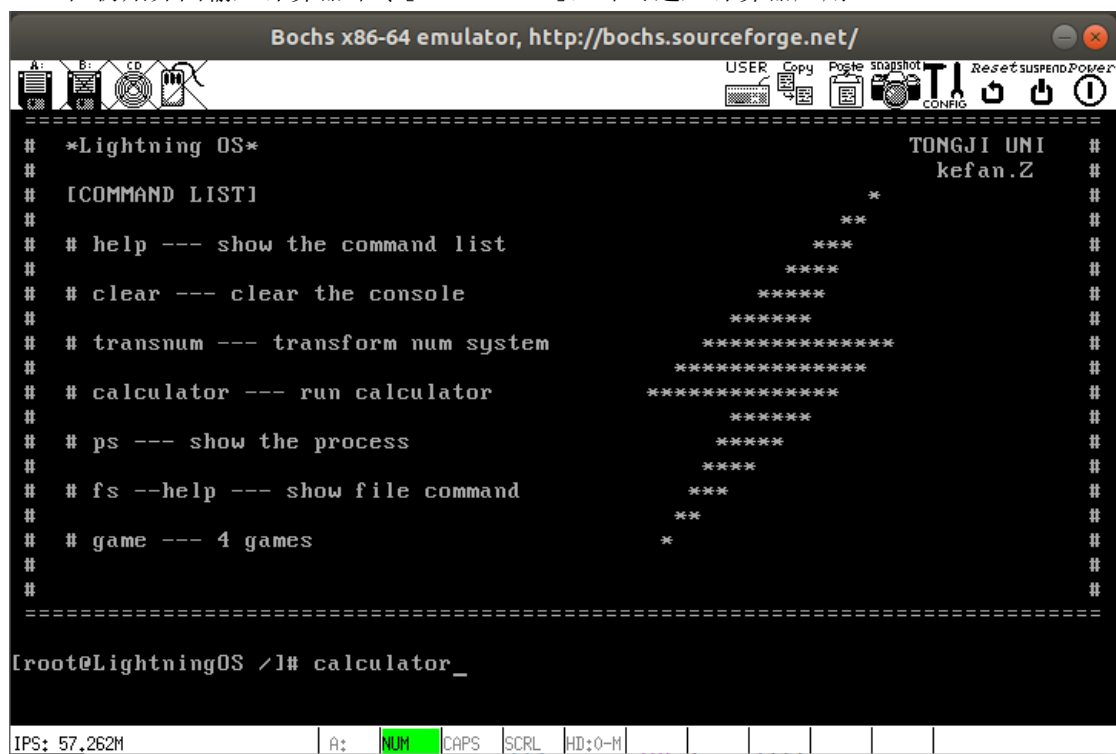
```
22.   myFlag = myChooseFlag[0]-48;
23.   start(myFlag);
24.   printf("OK!Game Start!\n\n");
25.
26.   if(myFlag == 1)
27.   {
28.       turn();
29.   }
30.   else if (myFlag == 2)
31.   {
32.       printBoard();
33.       printf("It's your turn.\n\n");
34.   }
35.
36.   }
37.   else if (strcmp(tag, PLACE) == 0)
38.   {
39.       printf("Please enter your steps:");
40.       char myStep[128];
41.       int len = read(*fd_stdin, myStep, sizeof(myStep));
42.       int whichstep = 0;
43.       command.numStep = myStep[0]-48;
44.       for(int i=1;i<len;i++)
45.       {
46.           if(myStep[i] == ',')
47.           {
48.               command.x[whichstep] = myStep[i-1] - 48;
49.               command.y[whichstep] = myStep[i+1] - 48;
50.               whichstep++;
51.           }
52.       }
53.       place(command, 2);
54.       printBoard();
55.       milli_delay(10000);
56.       turn();
57.   }
58.   }
59. }
```

2.2 复杂用户级应用

2.2.1 计算器

2.2.1.1 界面展示

在初始界面输入计算器命令[calculator]，即可进入计算器应用。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

=====
# *Lightning OS*                                     TONGJI UNI #
#                                                     kefan.Z  #
# [COMMAND LIST]                                     *           #
#                                                     **          #
# # help --- show the command list                   ***         #
#                                                     ****        #
# # clear --- clear the console                      *****      #
#                                                     ****          #
# # transnum --- transform num system                 ****          #
#                                                     ****          #
# # calculator --- run calculator                     ****          #
#                                                     ****          #
# # ps --- show the process                          ****          #
#                                                     ****          #
# # fs --help --- show file command                  ***          #
#                                                     **           #
# # game --- 4 games                                *            #
#                                                     #            #
#=====

[root@LightningOS /]# calculator_

=====
IPS: 57.262M | A: NUM | CAPS | SCRL | HD: 0-M |
```

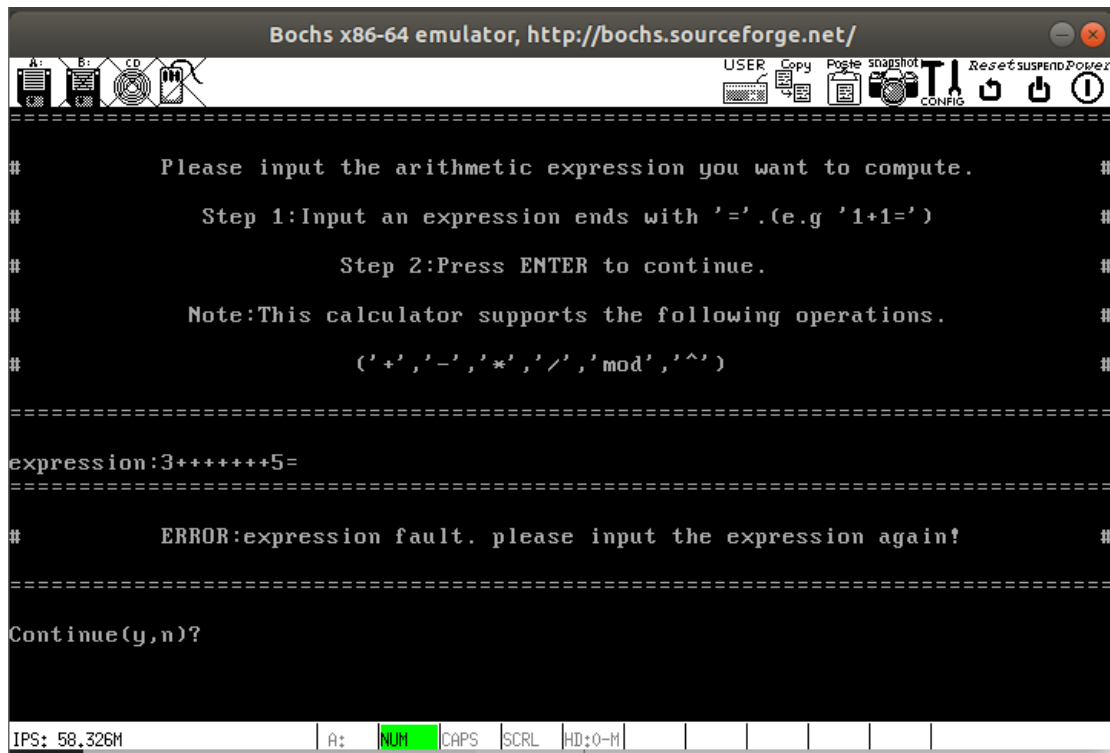
计算器应用上方首先给出了计算器的使用方法及支持的操作。该计算器支持加、减、乘、除、取余、乘方等 6 种运算。用户只需输入表达式，在表达式后添加 '=' 符号并按下 ENTER 键，系统便会自动进行计算，给出答案。同时，该计算器也支持表达式检查功能，能检查出符号错误、除 0 等表达式错误。



除零错误。



表达式符号出错。



2.2.1.2 核心代码

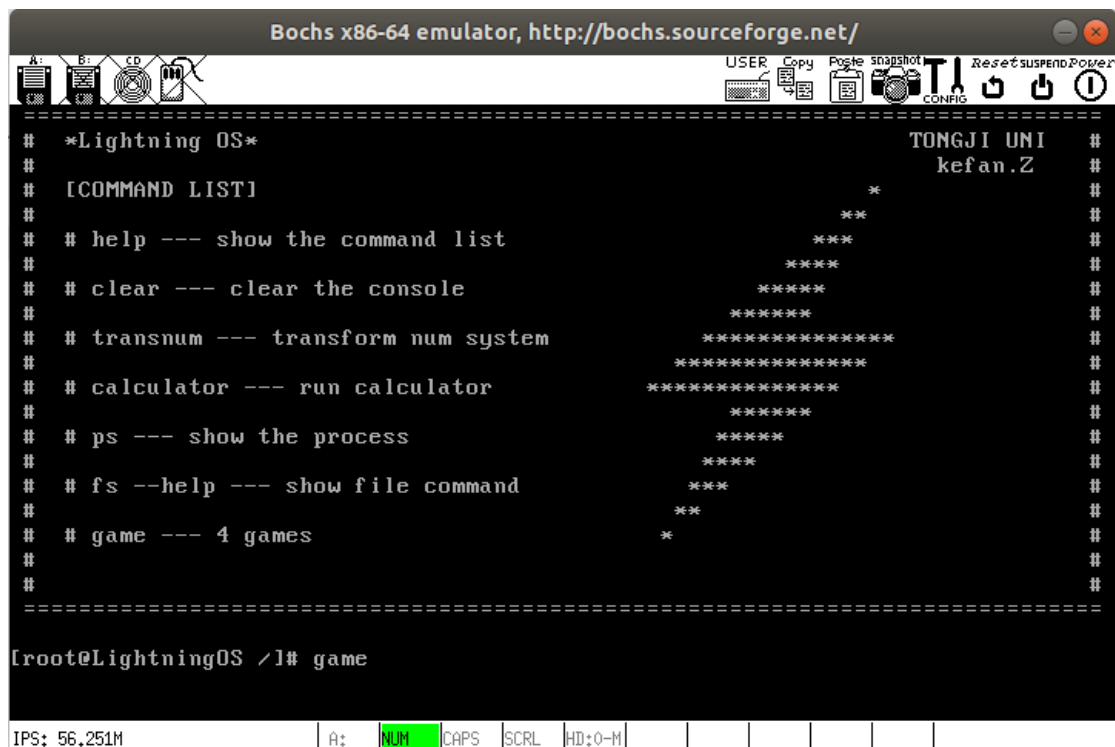
```
1. //初始化
2. void Calculator_Initialize(struct Calculator** this, int* fd_stdin)
3. {
4.     Calculator_Run(this, fd_stdin);
5.     Calculator_Clear(this);
6.     char option = '#'; int iscontinue = 1;
7.     while (iscontinue)
8.     {
9.         printf("Continue(y,n)?");
10.         char buff[2];
11.         int r = read(*fd_stdin, buff, 512);
12.         printf("\n");
13.         switch (buff[0])
14.         {
15.             case 'y':clear();Calculator_Run(this, fd_stdin); Calculator_Clear(this); break;
16.             case 'n':iscontinue = 0; break;
17.         }
18.     }
19. }
```

```
1. void Calculator_Run(struct Calculator** this, int* fd_stdin)
2. {
3.     Calculator_Input(this, fd_stdin);
4.     // 获得后缀表达式
5.     struct Node fro = {0,0,'#',0,0,NULL};
6.     struct Queue tmppostexpr = {&fro,&fro,1};
7.     struct Queue* postexpr = &tmppostexpr;
8.     Queue_Postfix(&postexpr, (*this)->expr->front);
9.
10.    struct Node* current = postexpr->front->next;
11.    while (current != NULL)
12.    {
13.        // 操作数进栈
14.        if (current->type == 0)
15.        {
16.            Stack_Push(&((*this)->box), current);
17.            current = current->next;
18.        }
19.        // 操作符运算
20.        if (current->type == 1)
21.        {
22.            if (current->charnum == '#')
23.            {
24.                current = current->next;
25.                continue;
26.            }
27.            if (Calculator_Calculate(this,&current)) == false)
28.            {
29.                return;
30.            };
31.        }
32.    }
33.    printf("Result:%d",(*this)->box->top->intnum);
34.    printf("\n\n");
35.};
```

2.2.2 贪吃蛇

2.2.2.1 界面展示

控制台输入[game]命令，控制台将会给出游戏列表。

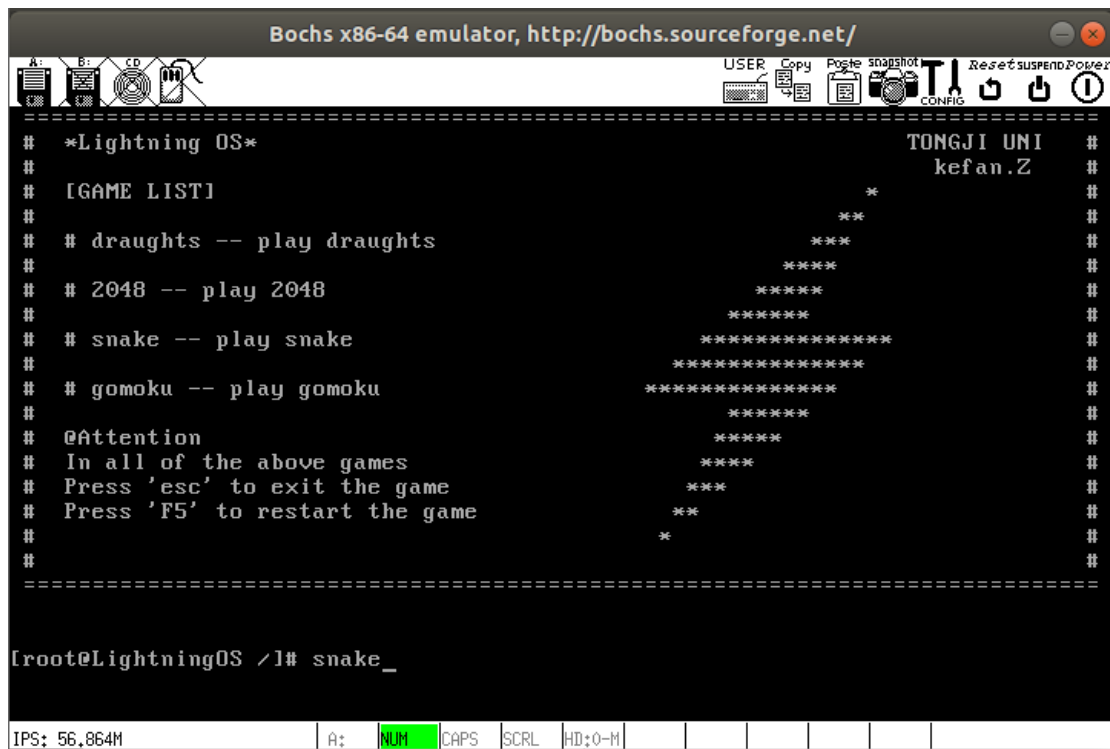


The screenshot shows the Bochs x86-64 emulator window. The title bar reads "Bochs x86-64 emulator, http://bochs.sourceforge.net/". The window contains a terminal window with the following content:

```
=====
# *Lightning OS*                                TONGJI UNI #
#                                                    kefan.Z  #
# [COMMAND LIST]                                *          #
#                                                    **         #
# # help --- show the command list                ***        #
#                                                    ****       #
# # clear --- clear the console                    *****    #
#                                                    ****        #
# # transnum --- transform num system              ****        #
#                                                    ****        #
# # calculator --- run calculator                  ****        #
#                                                    ****        #
# # ps --- show the process                       ****        #
#                                                    ****        #
# # fs --help --- show file command                ***         #
#                                                    **          #
# # game --- 4 games                              *           #
#                                                    #           #
#=====
[root@LightningOS /]# game
```

At the bottom of the window, there is a status bar with the following information: IPS: 56.251M, A: NUM, CAPS, SCRL, HD: 0-M, and several empty boxes.

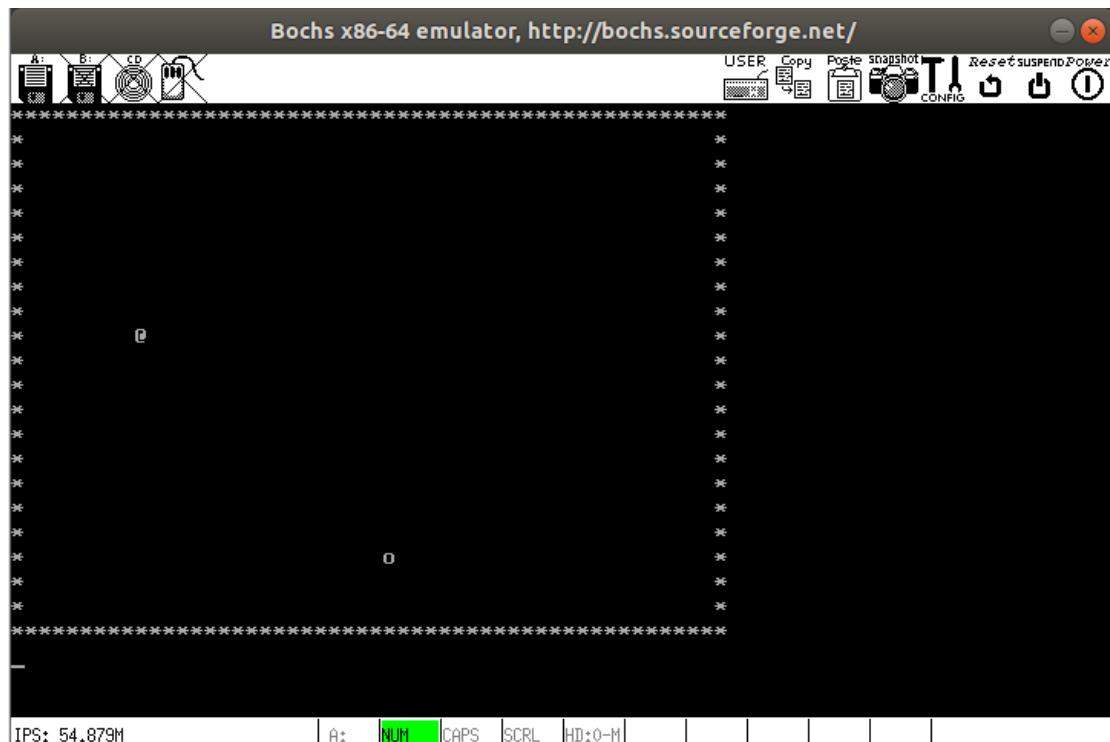
游戏列表界面给出 4 个对应的游戏以及启动游戏的命令。同时，在列表下方也给出了 4 个游戏通用的一些操作，比如说按 f5 键重置游戏，按 esc 键退出游戏。



输入贪吃蛇启动命令[snake]后，首先进入欢迎界面。由于贪吃蛇启动后，蛇会不停地移动，为了保证玩家已做好游戏的准备，玩家需要按下 ENTER 键以正式开始游戏。



贪吃蛇游戏界面标出了游戏边界、蛇身以及食物三个物体。与经典贪吃蛇游戏一致，玩家可以通过上下左右按键控制蛇的移动方向，吃到食物后蛇身会变长，一旦蛇头碰撞到边界或自身都算游戏失败。



2.2.2.2 核心代码

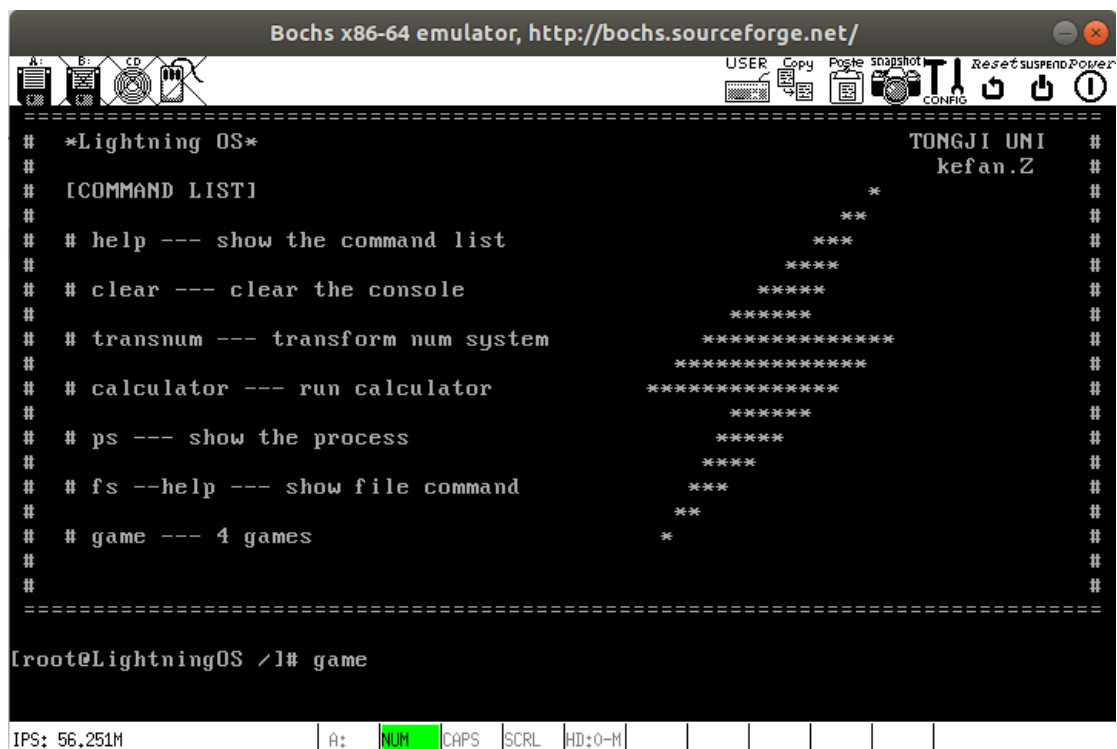
```
1. welcomeView_snake();
2.     while(iscontinue)
3.     {
4.         snake_init(map, snake); // 初始化地图
5.         while (1)
6.         {
7.             clear();
8.             milli_delay(100);
9.             move(snake, direct); // 让蛇开始移动
10.            if (!food[0] && !food[1]) { // 食物已经被吃掉--重新生成食物
11.                makeFood(food); // 生成食物坐标
12.            }
13.            makeMap(map, snake, food);
14.            showView();
15.            if (ifBump(head)) {
16.                printf("Game Over, your final score is: %d\n", head);
17.                printf("Press esc/f5 to exit/restart.\n");
18.                iscontinue = -1;
19.                while(iscontinue == -1){}
20.                break;
```

```
21.      }  
22.      }  
23.      }
```

2.2.3 游戏 2048

2.2.3.1 界面展示

控制台输入[game]命令，控制台将会给出游戏列表。



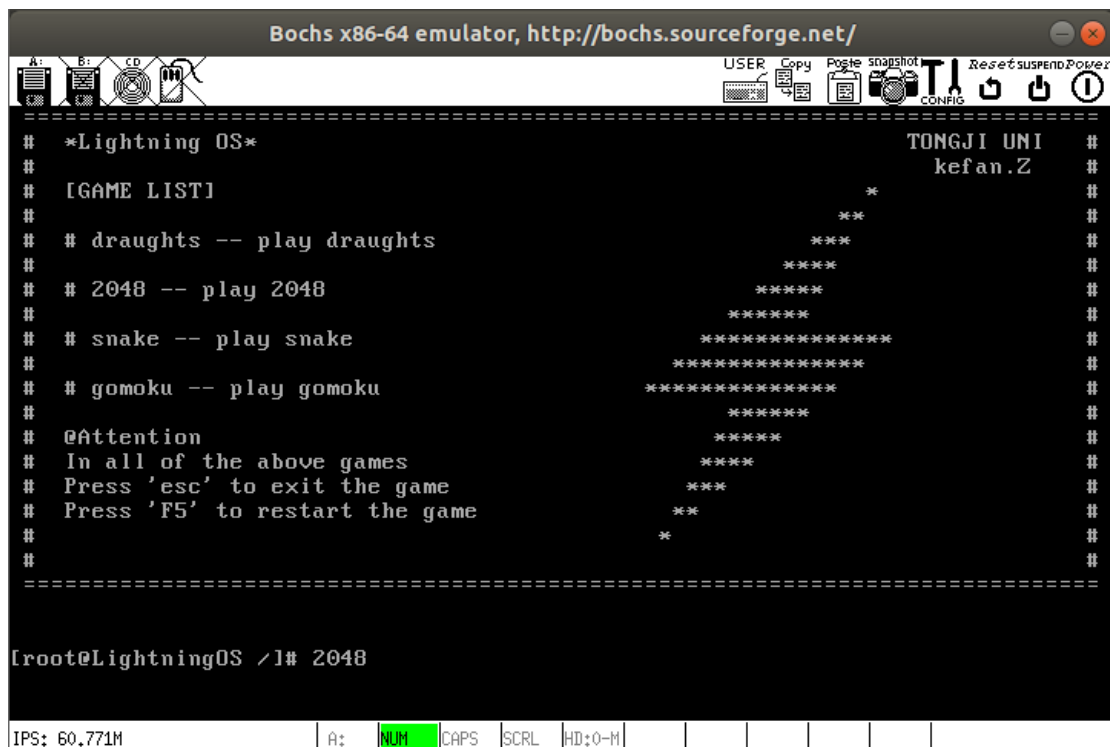
The screenshot shows the Bochs x86-64 emulator window. The title bar reads "Bochs x86-64 emulator, http://bochs.sourceforge.net/". The window contains a terminal window with the following content:

```
# *Lightning OS*
#
# [COMMAND LIST]
#
# # help --- show the command list
#
# # clear --- clear the console
#
# # transnum --- transform num system
#
# # calculator --- run calculator
#
# # ps --- show the process
#
# # fs --help --- show file command
#
# # game --- 4 games
#
=====

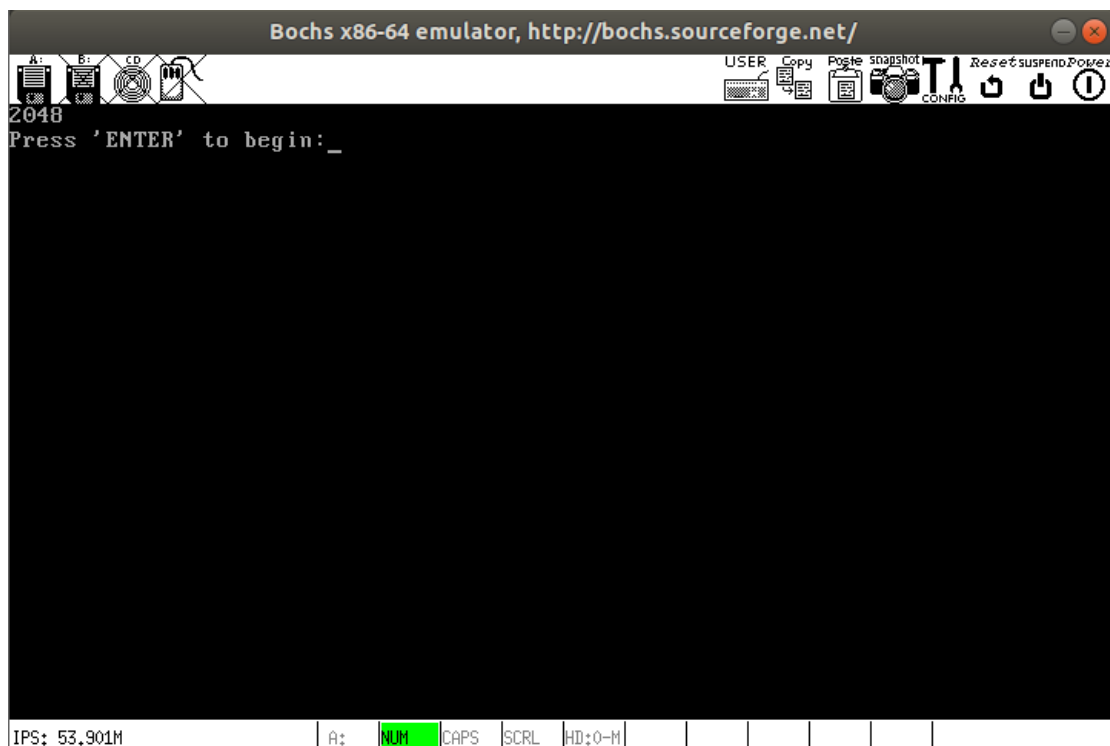
[root@LightningOS /]# game
```

The terminal output is decorated with asterisks and the text "TONGJI UNI" and "kefan.Z". The Bochs window also shows a toolbar with icons for USER, Copy, Paste, Snapshot, CONFIG, Reset, Suspend, and Power.

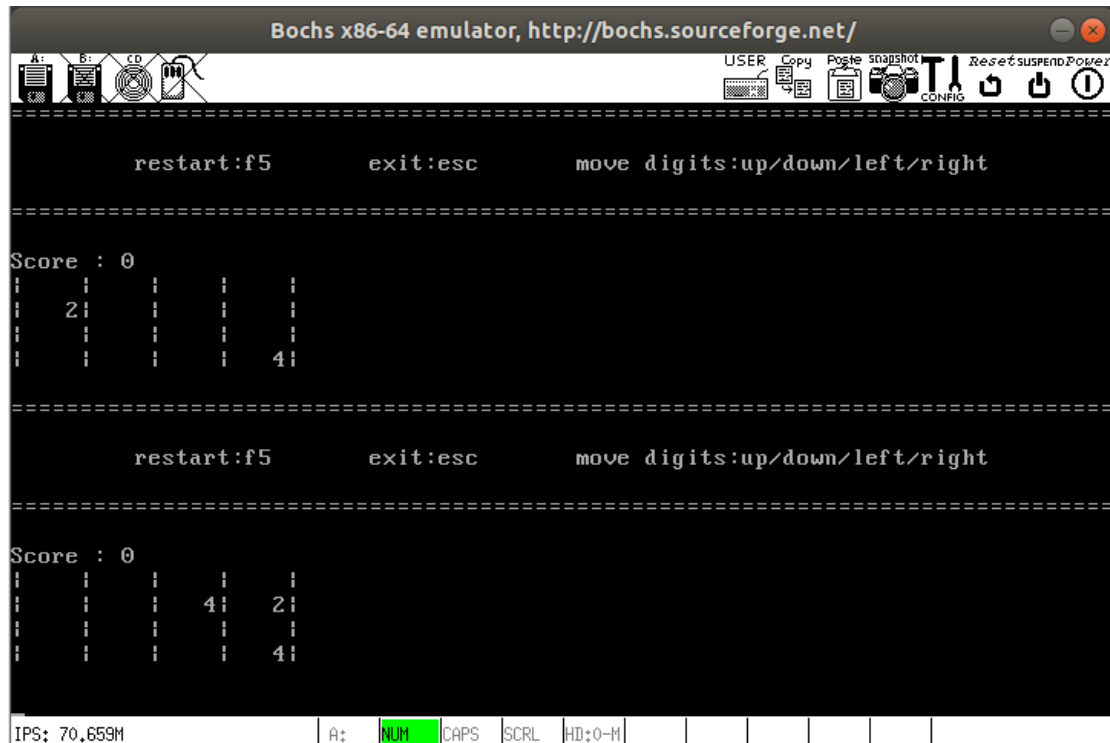
游戏列表界面给出 4 个对应的游戏以及启动游戏的命令。同时，在列表下方也给出了 4 个游戏通用的一些操作，比如说按 f5 键重置游戏，按 esc 键退出游戏。



输入 2048 游戏启动命令[2048]后，玩家进入游戏欢迎界面，玩家需按下 ENTER 键表示已准备好开始游戏。



与经典 2048 游戏一样，玩家可通过上下左右按键控制方块的移动，相同数字的方块碰撞后会产生合并，生成一个更大的数字。当盘面 16 个格子全满而已无法进行方块合并时，游戏结束。在整个游戏过程中，系统将会记录玩家的分数。



2.2.3.2 核心代码

```
1. init_2048();
2.     welcomeView();
3.     while(1)
4.     {
5.         if(checkAction == 0)
6.         {
7.             clear();
8.             init_2048();
9.             checkAction = 1;
10.        }
11.        else if(checkAction == -1)
12.        {
13.            break;
14.        }
15.        else if(checkAction == -2)
16.        {
17.            printf("=====\n");
18.            center("Sorry, You lost !");
19.            char str[13] = "Score : ";
```

```
20.         char sc[5];
21.         sprintf(sc,"%d",score);
22.         for(int i=8;i<13;i++)
23.         {
24.             str[i] = sc[i-8];
25.         }
26.         printf("Final ");
27.         center(str);
28.
29.         unsigned short x,y;
30.         for (x=0; x<4; x++)
31.         {
32.             for(y = 0; y < 4; y++)
33.             {
34.                 printf("|");
35.                 printf("%s", cell[x][y]);
36.             }
37.             printf("|\n");
38.         }
39.         center("");
40.         center("Press esc/f5 to exit/restart.");
41.         while(checkAction == -2){}
42.     }
43.     else if(checkAction == 1)
44.     {
45.         if(displaychanged == 1 )
46.         {
47.             display();
48.             displaychanged = 0;
49.         }
50.     }
51. }
```

2.3 系统级应用

2.3.1 进程管理

2.3.1.1 界面展示

用户在初始界面输入命令[ps]后,即可查看当前正在运行的进程信息。其中包括进程 ID、进程名字、进程优先级、父进程 ID、进程创建用户等内容。

```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
# # ps --- show the process
# # fs --help --- show file command
# # game --- 4 games
#
=====
[root@LightningOS /]# ps
-----
PID      | name      | priority | parPID   | user
-----
0        | TTY       | 15       | 57       | root
1        | SYS       | 15       | 57       | root
2        | HD        | 15       | 57       | root
3        | FS        | 15       | 57       | root
4        | MM        | 15       | 57       | root
5        | TestA     | 5        | 57       | root
6        | TestB     | 5        | 57       | root
7        | TestC     | 5        | 57       | root
-----
[root@LightningOS /]#
```

2.3.1.2 核心代码

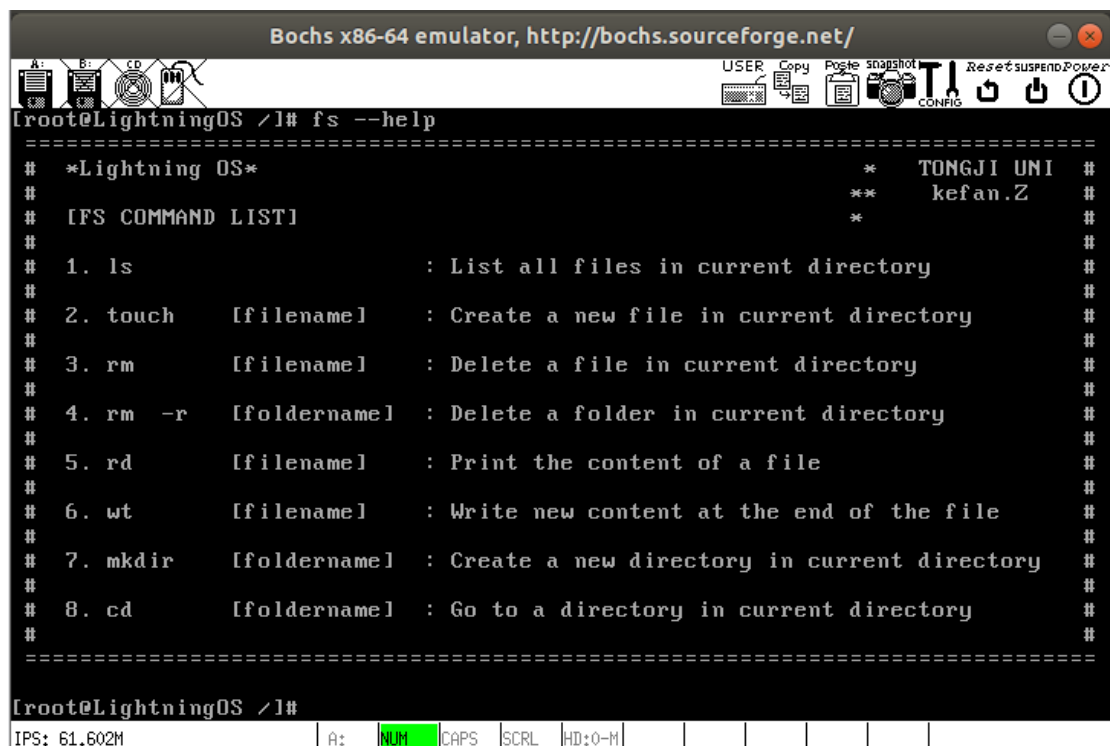
```
1. void processManage()
2. {
3.     int i;
4.     printf("-----\n");
5.     printf("%9s      |%10s      |%12s      |%9s      |%9s      \n", "PID", "name", "priority", "parPID", "user");
6.     printf("-----\n");
7.     for (i = 0; i < NR_TASKS + NR_PROCS; ++i)
8.     {
9.         // 读取 proc_table 数组获取进程信息
10.        if (proc_table[i].p_flags != FREE_SLOT)
11.            printf("%8d      |%10s      |%8d      |%8d      |%s\n",
```

```
12.         proc_table[i].pid, proc_table[i].name, proc_table[i].priority, proc_table[
            i].p_parent, proc_table[i].user);
13.     }
14.     printf("-----\n\n");
15. }
```

2.3.2 文件系统

2.3.2.1 界面展示

在初始界面输入文件系统帮助命令[fs --help]后，系统将会列出文件系统的一系列操作命令。其中包括查看当前目录下的文件、创建文件、创建文件夹、移除文件、移除文件夹、读文件、写文件、进入文件夹等功能。



创建文件夹：输入 `mkdir [foldername]` 创建名为 `foldername` 的文件夹

```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
=====
# *Lightning OS*                                     TONGJI UNI #
#                                                     kefan.Z   #
# [COMMAND LIST]                                     *           #
#                                                     **          #
# # help --- show the command list                 ***         #
#                                                     ****        #
# # clear --- clear the console                    *****       #
#                                                     ****         #
# # transnum --- transform num system              ****          #
#                                                     ****           #
# # calculator --- run calculator                  ****            #
#                                                     ****             #
# # ps --- show the process                       ****              #
#                                                     ****               #
# # fs --help --- show file command               ***                #
#                                                     **                 #
# # game --- 4 games                             *                  #
#                                                     *                   #
#=====
[root@LightningOS /]# mkdir osdesign
[root@LightningOS /]# _
```

查看当前目录下的文件：输入 `ls` 查看当前目录下的文件夹和文件。

可以看到名为 osdesign 的文件夹已创建成功。

Bochs x86-64 emulator, <http://bochs.sourceforge.net/>

USER Copy Paste Snapshot CONFIG Reset suspend Power

```

#
# # calculator --- run calculator
#
# # ps --- show the process
#
# # fs --help --- show file command
#
# # game --- 4 games
#
=====

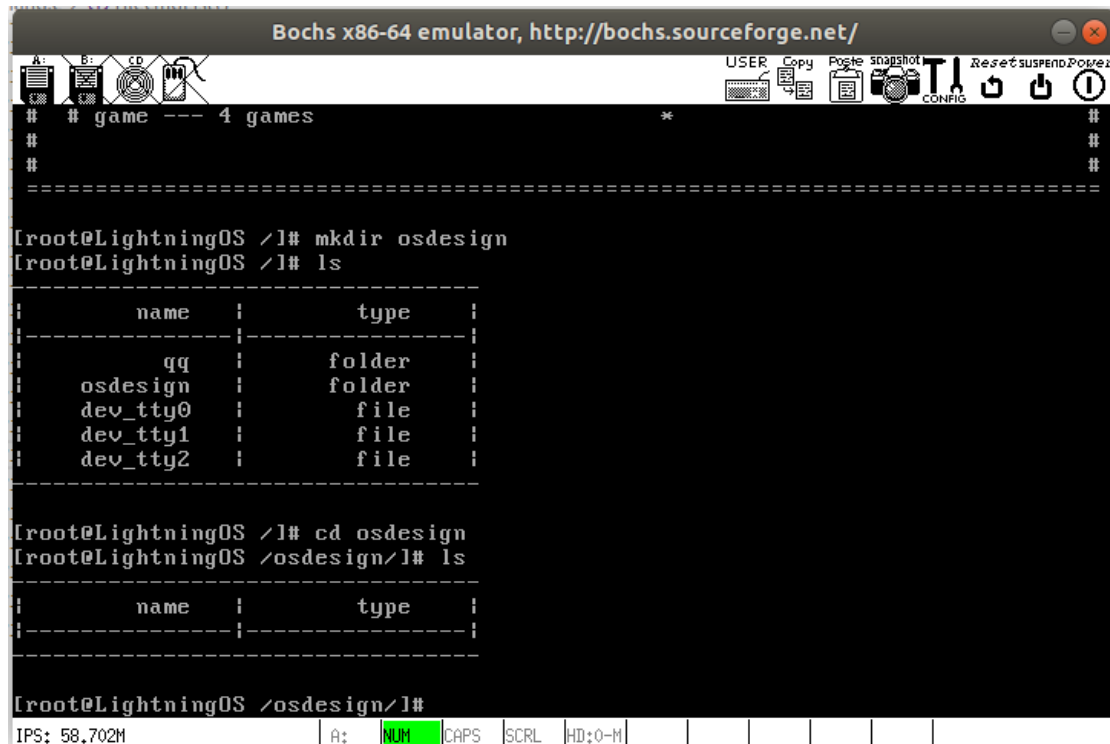
[root@LightningOS /]# mkdir osdesign
[root@LightningOS /]# ls
-----
|          name          |          type          |
|-----|-----|
|          qq            |          folder        |
|          osdesign       |          folder        |
|          dev_tty0       |          file          |
|          dev_tty1       |          file          |
|          dev_tty2       |          file          |
|-----|-----|

[root@LightningOS /]#

```

TPS: 58.047M NUM CAPS SCRI HD:0-M

进入文件夹：输入 `cd [foldername]` 进入文件夹。此处进入 `osdesign` 文件夹，可以看见路径名已经改变，内部也没有任何文件。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

# # game --- 4 games
#
=====

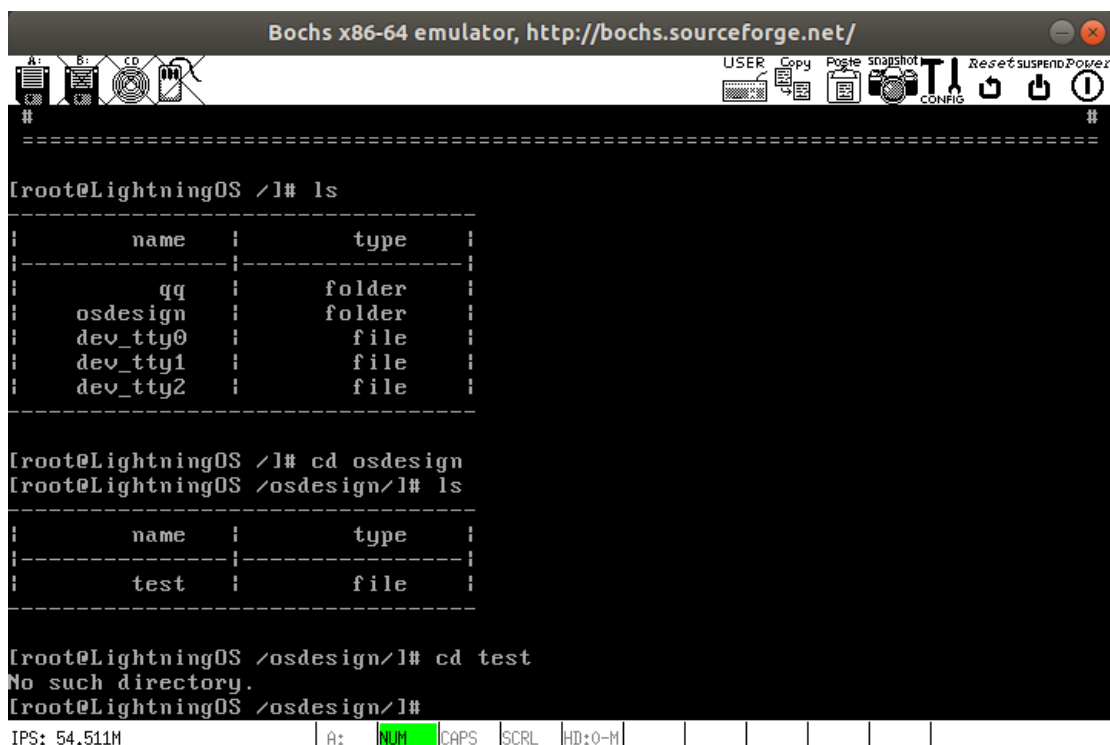
[root@LightningOS /]# mkdir osdesign
[root@LightningOS /]# ls
-----
| name | type |
-----
| qq | folder |
| osdesign | folder |
| dev_tty0 | file |
| dev_tty1 | file |
| dev_tty2 | file |
-----

[root@LightningOS /]# cd osdesign
[root@LightningOS /osdesign/]# ls
-----
| name | type |
-----
-----

[root@LightningOS /osdesign/]#

IPS: 58.702M | A: NUM | CAPS | SCRL | HD:0-M | | | | | | | |
```

若用户输入不规范，企图 `cd` 一个文件，则文件系统会提示没有这样的目录。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

#
=====

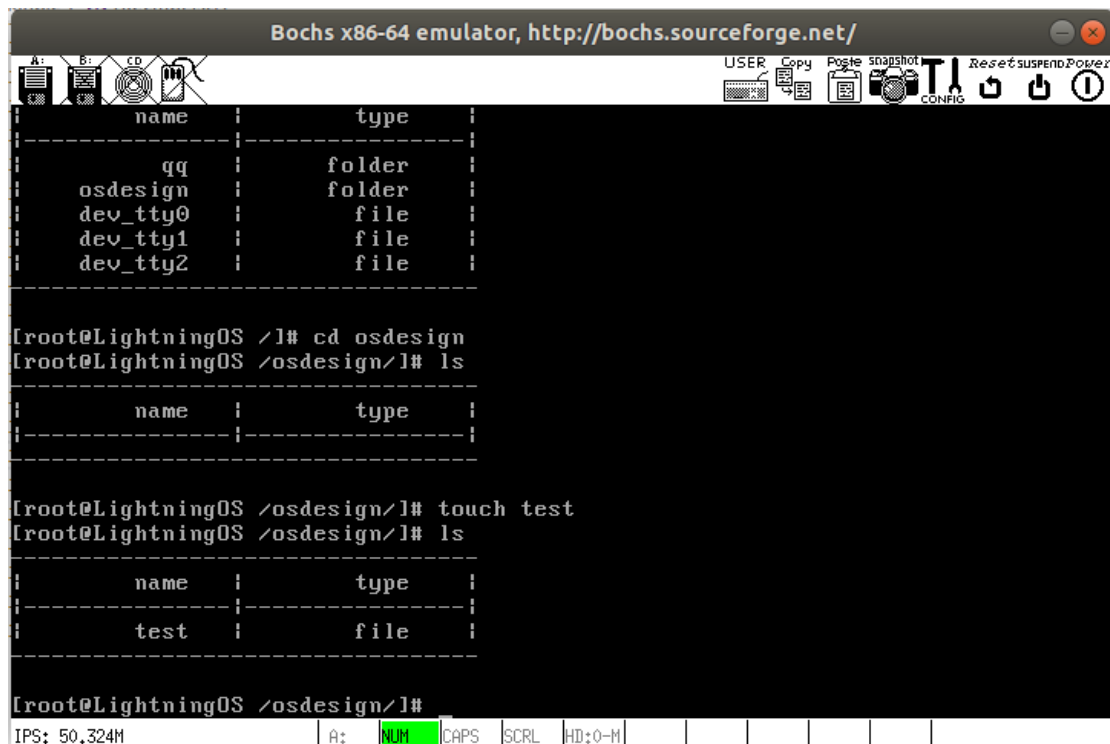
[root@LightningOS /]# ls
-----
| name | type |
-----
| qq | folder |
| osdesign | folder |
| dev_tty0 | file |
| dev_tty1 | file |
| dev_tty2 | file |
-----

[root@LightningOS /]# cd osdesign
[root@LightningOS /osdesign/]# ls
-----
| name | type |
-----
| test | file |
-----

[root@LightningOS /osdesign/]# cd test
No such directory.
[root@LightningOS /osdesign/]#

IPS: 54.511M | A: NUM | CAPS | SCRL | HD:0-M | | | | | | | |
```

创建文件：输入 `touch [filename]` 创建文件。此处创建了一个名为 `test` 的文件，可以看见文件创建成功。



The screenshot shows the Bochs x86-64 emulator window. The terminal displays the following commands and output:

```
[root@LightningOS /]# cd osdesign
[root@LightningOS /osdesign/]# ls
```

name	type
qq	folder
osdesign	folder
dev_tty0	file
dev_tty1	file
dev_tty2	file

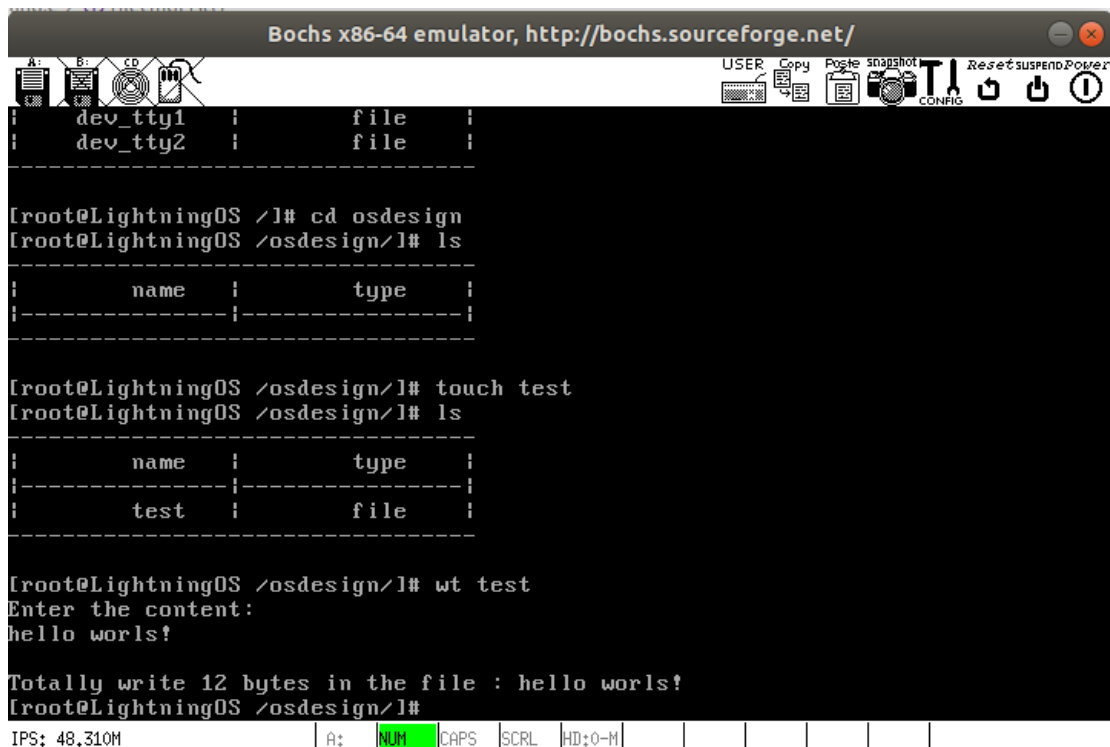
```
[root@LightningOS /osdesign/]# touch test
[root@LightningOS /osdesign/]# ls
```

name	type
test	file

```
[root@LightningOS /osdesign/]#
```

The status bar at the bottom shows: IPS: 50.324M, A: NUM, CAPS, SCRL, HD: 0-M.

写文件：输入 `wt [filename]` 对文件进行修改。此处在 `test` 文件中写入一句 `hello world!`。



The screenshot shows the Bochs x86-64 emulator window. The terminal displays the following commands and output:

```
[root@LightningOS /]# cd osdesign
[root@LightningOS /osdesign/]# ls
```

name	type
dev_tty1	file
dev_tty2	file

```
[root@LightningOS /osdesign/]# touch test
[root@LightningOS /osdesign/]# ls
```

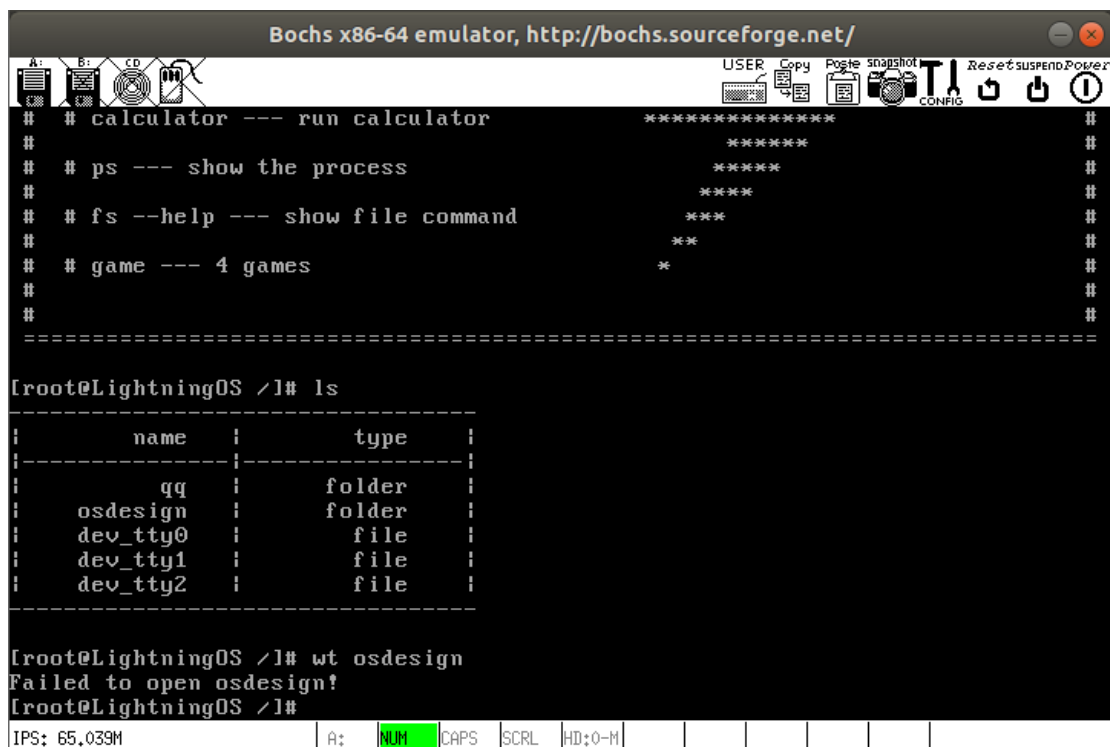
name	type
test	file

```
[root@LightningOS /osdesign/]# wt test
Enter the content:
hello worls!

Totally write 12 bytes in the file : hello worls!
[root@LightningOS /osdesign/]#
```

The status bar at the bottom shows: IPS: 48.310M, A: NUM, CAPS, SCRL, HD: 0-M.

若用户输入不规范，企图 wt 一个文件夹目录的话，文件系统会提示用户打开失败！



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

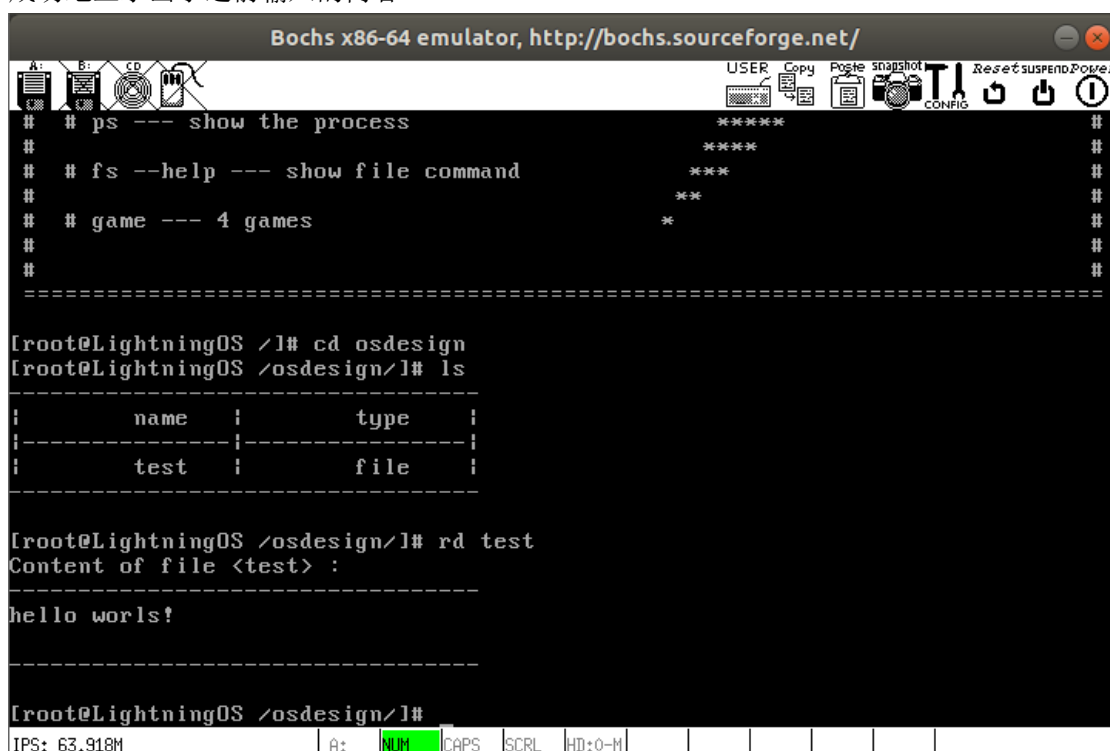
# # calculator --- run calculator
# # ps --- show the process
# # fs --help --- show file command
# # game --- 4 games
#
=====

[root@LightningOS /]# ls
-----
|          name          |          type          |
|-----|-----|
|          qq            |          folder        |
|          osdesign       |          folder        |
|          dev_tty0       |          file          |
|          dev_tty1       |          file          |
|          dev_tty2       |          file          |
|-----|-----|

[root@LightningOS /]# wt osdesign
Failed to open osdesign!
[root@LightningOS /]#

IPS: 65.039M  A: NUM CAPS SCRL HD: 0-M
```

读文件：输入 rd [filename] 读取相应文件中的内容。此处读取了 test 文件中的内容，成功地显示出了之前输入的内容。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

# # ps --- show the process
# # fs --help --- show file command
# # game --- 4 games
#
=====

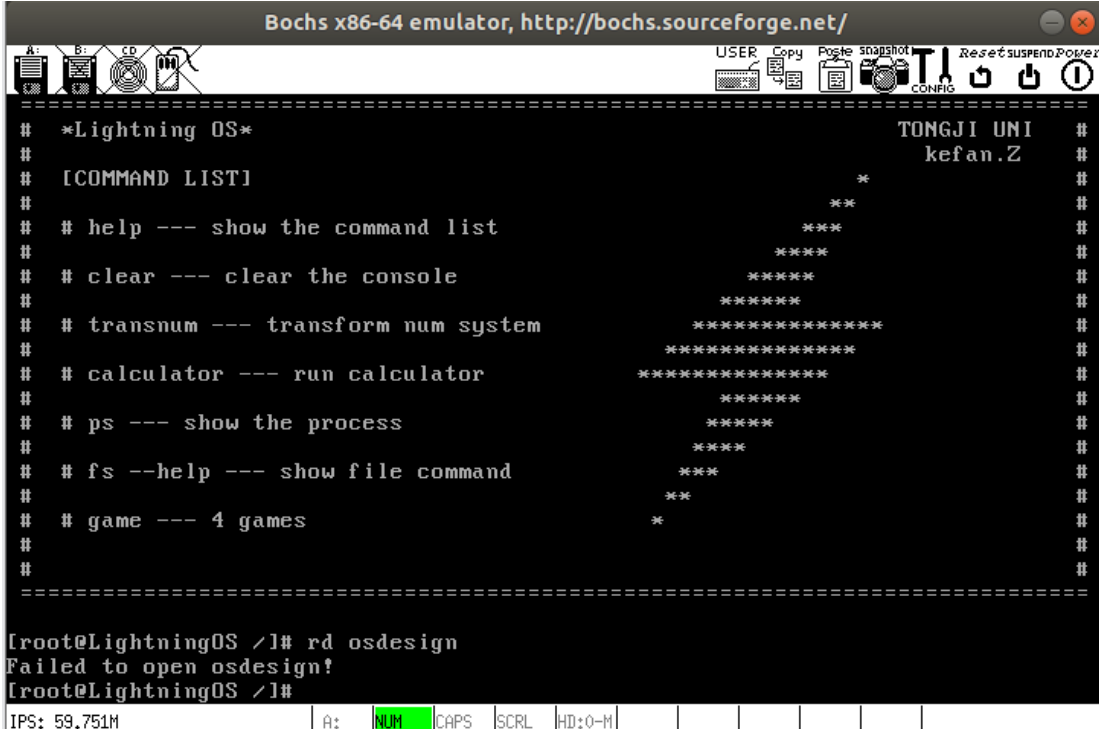
[root@LightningOS /]# cd osdesign
[root@LightningOS /osdesign/]# ls
-----
|          name          |          type          |
|-----|-----|
|          test          |          file          |
|-----|-----|

[root@LightningOS /osdesign/]# rd test
Content of file <test> :
-----
hello worls!
-----

[root@LightningOS /osdesign/]#

IPS: 63.918M  A: NUM CAPS SCRL HD: 0-M
```

若用户输入不规范，企图 rd 一个文件夹目录，文件系统将会提示无法打开文件。

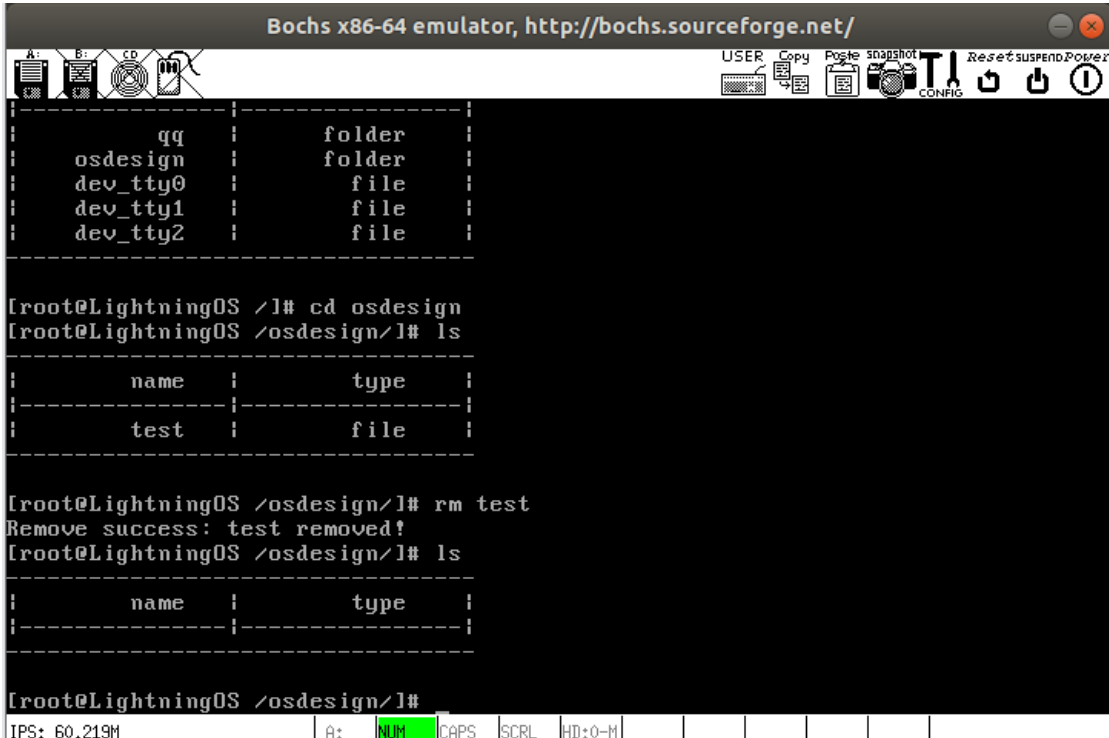


```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

# *Lightning OS*
# [COMMAND LIST]
# # help --- show the command list
# # clear --- clear the console
# # transnum --- transform num system
# # calculator --- run calculator
# # ps --- show the process
# # fs --help --- show file command
# # game --- 4 games

[root@LightningOS /]# rd osdesign
Failed to open osdesign!
[root@LightningOS /]#
```

移除文件：输入 rm [filename] 删除对应的文件。此处删除了文件 test，文件系统提示文件删除成功，文件也确实删除了。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

qq      folder
osdesign  folder
dev_tty0 file
dev_tty1 file
dev_tty2 file

[root@LightningOS /]# cd osdesign
[root@LightningOS /osdesign/]# ls

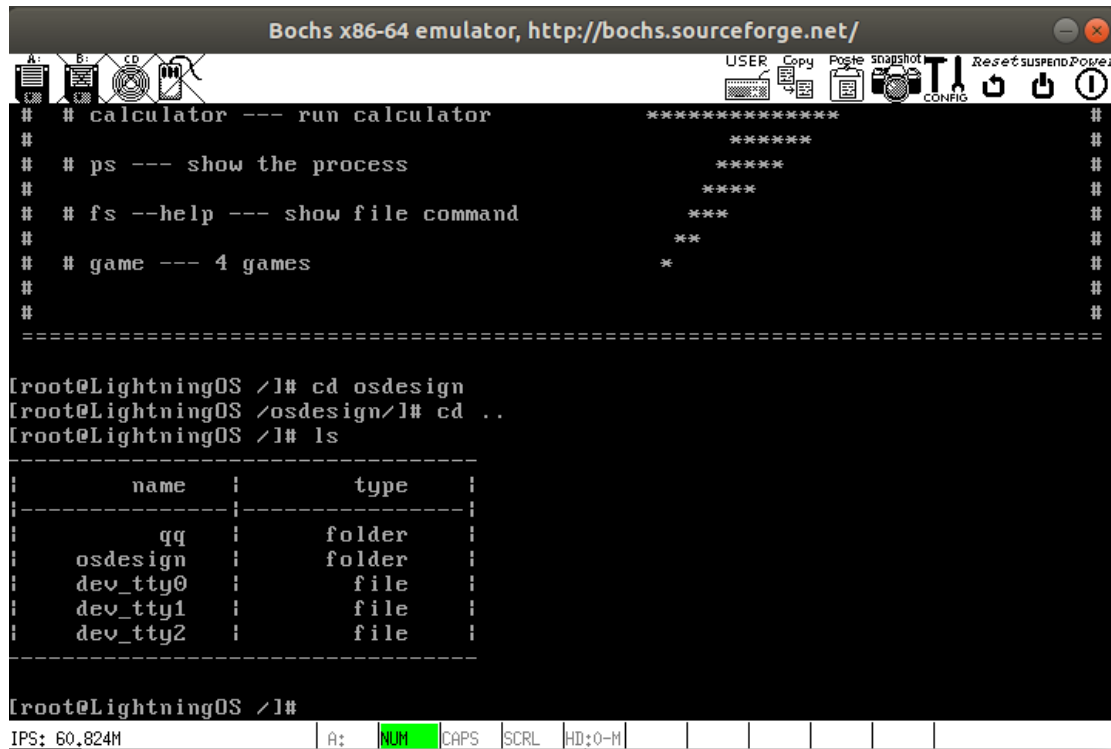
name      type
test      file

[root@LightningOS /osdesign/]# rm test
Remove success: test removed!
[root@LightningOS /osdesign/]# ls

name      type

[root@LightningOS /osdesign/]#
```

退出文件夹：输入 `cd ..` 退出当前目录，进入上一级目录。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

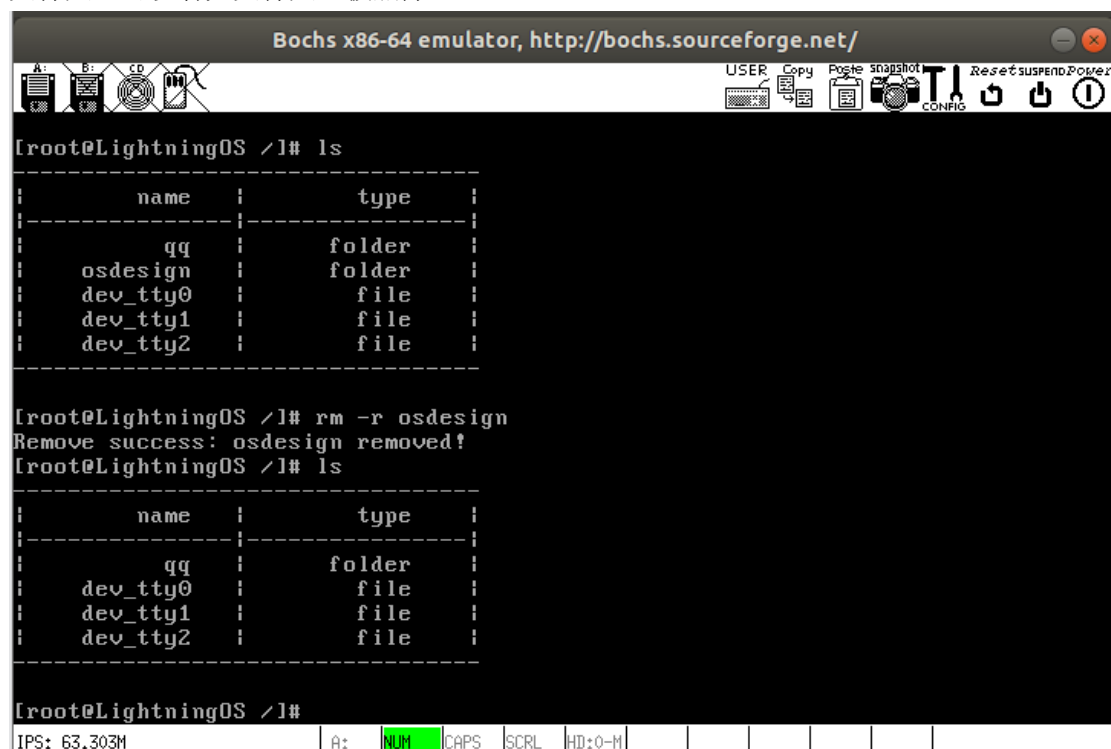
# # calculator --- run calculator
# # ps --- show the process
# # fs --help --- show file command
# # game --- 4 games
#
=====

[root@LightningOS /]# cd osdesign
[root@LightningOS /osdesign/]# cd ..
[root@LightningOS /]# ls
-----
|      name      |      type      |
|-----|-----|
|      qq      |      folder    |
|    osdesign    |      folder    |
| dev_tty0      |       file     |
| dev_tty1      |       file     |
| dev_tty2      |       file     |
|-----|-----|

[root@LightningOS /]#
```

IPS: 60.824M A: NUM CAPS SCRL HD:0-M

移除文件夹：输入 `rm -r [foldername]` 删除文件夹。此处删除了原先创建的 `osdesign` 文件夹，可以看见文件夹已被删除。



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/

[root@LightningOS /]# ls
-----
|      name      |      type      |
|-----|-----|
|      qq      |      folder    |
|    osdesign    |      folder    |
| dev_tty0      |       file     |
| dev_tty1      |       file     |
| dev_tty2      |       file     |
|-----|-----|

[root@LightningOS /]# rm -r osdesign
Remove success: osdesign removed!
[root@LightningOS /]# ls
-----
|      name      |      type      |
|-----|-----|
|      qq      |      folder    |
| dev_tty0      |       file     |
| dev_tty1      |       file     |
| dev_tty2      |       file     |
|-----|-----|

[root@LightningOS /]#
```

IPS: 63.303M A: NUM CAPS SCRL HD:0-M

2.3.2.2 核心代码

```
1. int parsecmd(char* rdbuf, char* curdir)
```

```
2. {
3.     char cmd[20];
4.     char filename1[128];
5.     char filename2[128];
6.     // 解析命令
7.     int pos = 0;
8.     while (rdbuf[pos] != ' ' && rdbuf[pos] != 0) // 读取指令
9.     {
10.        cmd[pos] = rdbuf[pos];
11.        pos++;
12.    }
13.    cmd[pos] = 0;
14.    if (rdbuf[pos] != 0) // 指令还未结束
15.    {
16.        pos++;
17.        int len = pos;
18.        while (rdbuf[pos] != ' ' && rdbuf[pos] != 0) // 读取第一个参数（文件名或指令属性）
19.        {
20.            filename1[pos - len] = rdbuf[pos];
21.            pos++;
22.        }
23.        filename1[pos - len] = 0;
24.    }
25.    else // 不存在第一个参数，将其置空
26.    {
27.        filename1[0] = 0;
28.    }
29.    if (rdbuf[pos] != 0) // 指令还未结束
30.    {
31.        pos++;
32.        int len = pos;
33.        while (rdbuf[pos] != ' ' && rdbuf[pos] != 0) // 读取第二个参数（操作由参时的文件名）
34.        {
35.            filename2[pos - len] = rdbuf[pos];
36.            pos++;
37.        }
38.        filename2[pos - len] = 0;
39.    }
40.    else // 不存在第二个参数，将其置空
41.    {
42.        filename2[0] = 0;
43.    }
```

```
44.  
45.    // 比对用户输入内容执行相应命令  
46.    // ls 展示当下目录所有文件与目录  
47.    // cd 进入目录  
48.    // mkdir 创建目录  
49.    // touch 创建普通文件  
50.    // wt 写文件  
51.    // rd 读文件  
52.    // rm 删除文件或目录  
53.    if (strcmp(cmd, "ls") == 0)  
54.    {  
55.        myls(curdir);  
56.        return 1;  
57.    }  
58.    else if (strcmp(cmd, "cd") == 0)  
59.    {  
60.        mycd(curdir, filename1);  
61.        return 1;  
62.    }  
63.    else if (strcmp(cmd, "mkdir") == 0)  
64.    {  
65.        mymkdir(curdir, filename1);  
66.        return 1;  
67.    }  
68.    else if (strcmp(cmd, "touch") == 0)  
69.    {  
70.        mytouch(curdir, filename1);  
71.        return 1;  
72.    }  
73.    else if (strcmp(cmd, "rm") == 0)  
74.    {  
75.        //0 means regular file, 1 means dir file  
76.        if (strcmp(filename1, "-r") == 0)  
77.        {  
78.            myrm(curdir, filename2, 1);  
79.        }  
80.        else {  
81.            myrm(curdir, filename1, 0);  
82.        }  
83.        return 1;  
84.    }  
85.    else if (strcmp(cmd, "wt") == 0)  
86.    {  
87.        mywt(curdir, filename1);
```

```
88.         return 1;
89.     }
90.     else if (strcmp(cmd, "rd") == 0)
91.     {
92.         myrd(curdir, filename1);
93.         return 1;
94.     }
95.     else if (strcmp(cmd, "mv") == 0)
96.     {
97.         mymv(curdir, filename1, filename2);
98.         return 1;
99.     }
100.    else
101.    {
102.        return 0;
103.    }
104.}
```


3 模块修改

3.1 进程调度模块

进程调度原先采用时间片轮转算法与优先级调度算法的一种结合，使优先级较高的进程在一定时间内能获得较多的时间片，用以处理进程。这样将会导致一个问题，对于一些处理时间较长而优先级又比较小的进程，其生命周期会很长，从运行态到就绪态的切换次数也会很多。而进程间的切换本身是占用一定资源和时间的，因此会造成一定程度的资源浪费。

```
int running_prio;  
int cut_num;
```

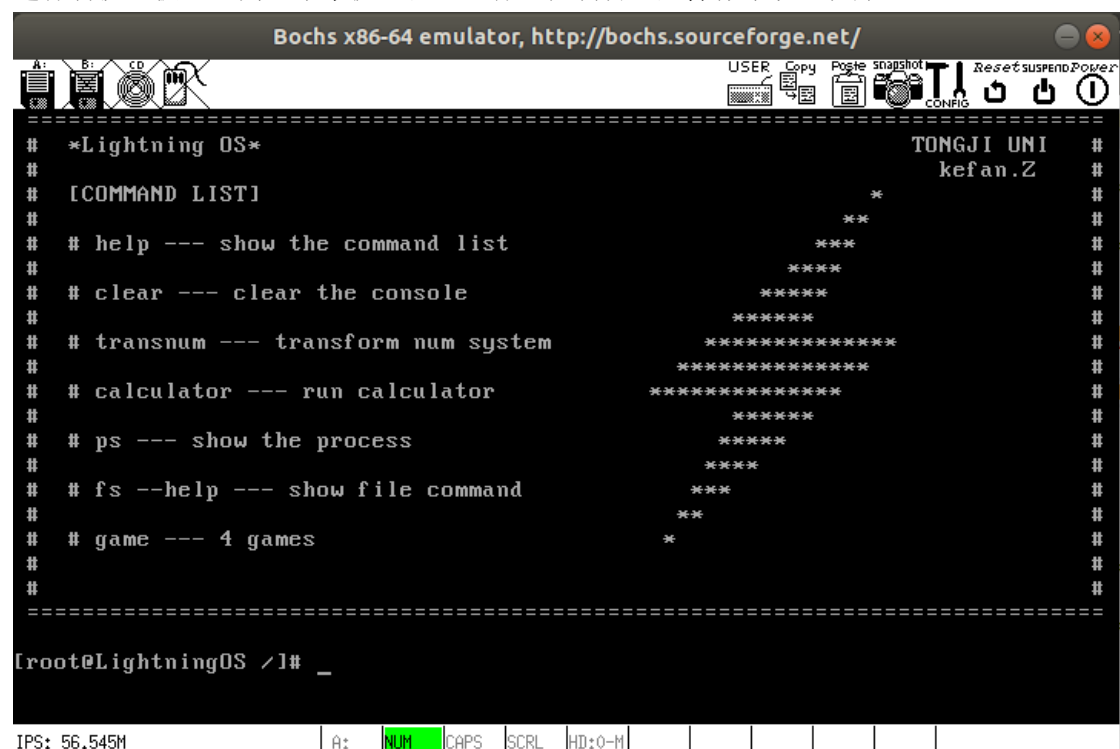
考虑到这种情况，在进程结构体中加入两个属性，running_prio 属性记录运行时优先级，其初值为自身优先级；cut_num 属性记录进程被切换的次数，其初值为 0。同时修改进程调度算法，每次进程被切换时，即使其切换次数属性自加，当切换次数达到一定阈值后（此处设定为 20 次），使其运行时优先级提高（此处设定为+3）。进程调度算法每次会比较所有进程的运行时优先级，并挑选其中最高的一个进行执行。

```
1. PUBLIC void schedule()  
2. {  
3.     struct proc* p;  
4.     int greatest_ticks = 0;  
5.  
6.     while (!greatest_ticks)  
7.     {  
8.         int oripid = p_proc_ready->pid;  
9.         // 动态调整优先级  
10.        for (p = &FIRST_PROC; p <= &LAST_PROC; p++)  
11.        {  
12.            if (p->p_flags == 0)  
13.            {  
14.                if (p->cut_num >= 20)  
15.                {  
16.                    if(p->running_prio <= 30)  
17.                    {  
18.                        p->running_prio += 3;  
19.                    }  
20.                    p->cut_num = 0;  
21.                }  
22.            }  
23.        }  
24.        // 找出当前优先级最高的进程  
25.        for (p = &FIRST_PROC; p <= &LAST_PROC; p++)  
26.        {
```

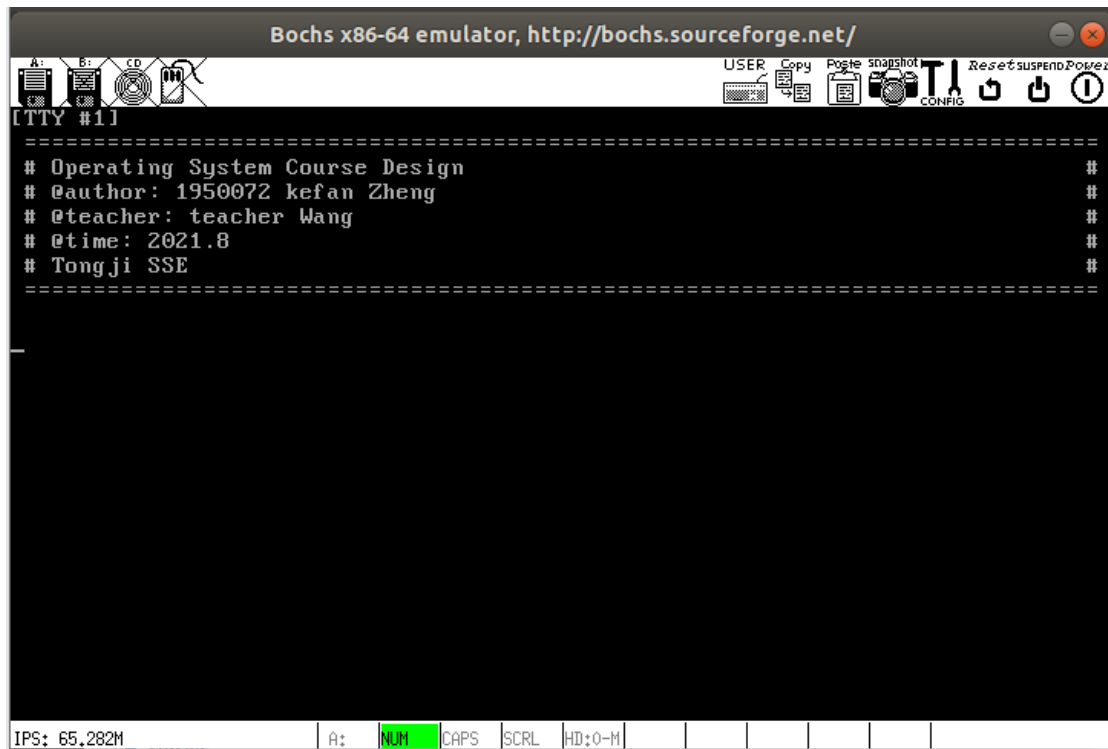
```
27.     if (p->p_flags == 0)
28.     {
29.         if (p->ticks > greatest_ticks)
30.         {
31.             greatest_ticks = p->ticks;
32.             p_proc_ready = p;
33.         }
34.     }
35. }
36. //进程切换时记录切换次数
37. if(oripid != p_proc_ready->pid)
38. {
39.     p_proc_ready->cut_num++;
40. }
41. //全部进程优先级为0，全部初始化
42. if (!greatest_ticks)
43.     for (p = &FIRST_PROC; p <= &LAST_PROC; p++)
44.         if (p->p_flags == 0)
45.             p->ticks = p->priority;
46. }
47. }
```

3.2 多控制台模块

将三个控制台分别绑定至 TestA、TestB、TestC 三个进程上，可通过键盘 Alt+F1/F2/F3 进行切换。彼此之间互不干扰，独立运行。控制台 1 是操作系统主程序。



控制台 2 是操作系统的项目简介。



控制台 3 打印 hello world!。



3.3 输入输出系统

输入输出系统模块主要对输入模块进行了修改。因为该操作系统与用户交互最多的设备是键盘，因此对键盘扫描码的识别和函数调用做出修改，使其满足不同控制台、不同应用之间的功能需要。

主要修改有：

1. 判断 Alt 组合键，若为 Alt+F1/F2/F3，则切换控制台。
2. 通过 TESTTASK 全局变量判断当前控制台的当前任务，并根据识别到的上下左右方向键、F5 键和 ESC 键调用相应任务的函数，使其实现特定的功能。如 2048 游戏中，方向键控制方块的移动；贪吃蛇中，方向键控制蛇的移动。

```
1. PUBLIC void in_process(TTY* tty, u32 key)
2. {
3.     if (!(key & FLAG_EXT)) {
4.         put_key(tty, key);
5.     }
6.     else {
7.         int raw_code = key & MASK_RAW;
8.         switch(raw_code) {
9.             case ENTER:
10.                put_key(tty, '\n');
11.                if(TESTTASK == 1)
12.                {
13.                    changeifdrastart(1);
14.                }
15.                else if(TESTTASK == 2)
16.                {
17.                    changeif2048start(1);
18.                }
19.                else if(TESTTASK == 3)
20.                {
21.                    changeifsnakestart(1);
22.                }
23.                else if(TESTTASK == 4)
24.                {
25.                    changeifgomokustart(1);
26.                }
27.                break;
28.             case BACKSPACE:
29.                if(TESTTASK == 0)
30.                {
31.                    if(tty->console->cursor-1>=boundary)
32.                    {
33.                        w_copy(tty->console->cursor-1,tty->console->cursor,32);
```

```
34.     tty->console->cursor--;
35.     set_cursor(tty->console->cursor);
36. }
37. }
38. else
39. {
40.     put_key(tty, '\b');
41. }
42. break;
43. case UP:
44.     if ((key & FLAG_SHIFT_L) ||
45.         (key & FLAG_SHIFT_R)) { /* Shift + Up */
46.         scroll_screen(tty->console, SCR_DN);
47.     }
48.     else
49.     {
50.         if(TESTTASK == 2)
51.         {
52.             addRandomValue2048(doProcess(0));
53.         }
54.         else if(TESTTASK == 3)
55.         {
56.             getKeysnake(0);
57.         }
58.     }
59.     break;
60. case DOWN:
61.     if ((key & FLAG_SHIFT_L) ||
62.         (key & FLAG_SHIFT_R)) { /* Shift + Down */
63.         scroll_screen(tty->console, SCR_UP);
64.     }
65.     else
66.     {
67.         if(TESTTASK == 2)
68.         {
69.             addRandomValue2048(doProcess(2));
70.         }
71.         else if(TESTTASK == 3)
72.         {
73.             getKeysnake(2);
74.         }
75.     }
76.     break;
77. case LEFT:
```

```
78.     if(TESTTASK == 2)
79.     {
80.         addRandomValue2048(doProcess(3));
81.     }
82.     else if(TESTTASK == 3)
83.     {
84.         getKeysnake(3);
85.     }
86.     else if(TESTTASK == 0)
87.     {
88.         tty->console->cursor -= 1;
89.         set_cursor(tty->console->cursor);
90.     }
91.     break;
92. case RIGHT:
93.     if(TESTTASK == 2)
94.     {
95.         addRandomValue2048(doProcess(1));
96.     }
97.     else if(TESTTASK == 3)
98.     {
99.         getKeysnake(1);
100.    }
101.    else if(TESTTASK == 0)
102.    {
103.        tty->console->cursor += 1;
104.        set_cursor(tty->console->cursor);
105.    }
106.    break;
107. case ESC:
108.     if(TESTTASK == 1)
109.     {
110.
111.     }
112.     else if(TESTTASK == 2)
113.     {
114.         change_checkAction(-1);
115.     }
116.     else if(TESTTASK == 3)
117.     {
118.         changeiscontinue_snake(0);
119.     }
120.     else if(TESTTASK == 4)
121.     {
```

```
122.     gomexit();
123. }
124. break;
125. case F1:
126. case F2:
127.     if ((key & FLAG_ALT_L) ||
128.         (key & FLAG_ALT_R)) { /* Alt + F1~F12 */
129.         select_console(raw_code - F1);
130.     }
131.     break;
132. case F3:
133.     if ((key & FLAG_ALT_L) ||
134.         (key & FLAG_ALT_R)) { /* Alt + F1~F12 */
135.         select_console(raw_code - F1);
136.     }
137.     break;
138. case F4:
139.     if ((key & FLAG_ALT_L) ||
140.         (key & FLAG_ALT_R)) { /* Alt + F1~F12 */
141.         select_console(raw_code - F1);
142.     }
143.     break;
144. case F5:
145.     if ((key & FLAG_ALT_L) ||
146.         (key & FLAG_ALT_R)) { /* Alt + F1~F12 */
147.         select_console(raw_code - F1);
148.     }
149.     else
150.     {
151.         if(TESTTASK == 1)
152.         {
153.
154.         }
155.         else if(TESTTASK == 2)
156.         {
157.             change_checkAction(0);
158.         }
159.         else if(TESTTASK == 3)
160.         {
161.             changeiscontinue_snake(1);
162.         }
163.         else if(TESTTASK == 4)
164.         {
165.             gomrestart();
```

```
166.     }
167. }
168. break;
169. case F6:
170. case F7:
171. case F8:
172. case F9:
173. case F10:
174. case F11:
175. case F12:
176. if ((key & FLAG_ALT_L) ||
177.     (key & FLAG_ALT_R)) { /* Alt + F1~F12 */
178.     select_console(raw_code - F1);
179. }
180. break;
181. default:
182. break;
183. }
184. }
185. }
```