1.项目描述

1.1 项目目的

操作系统课程设计旨在让我们通过学习并实现一个简单而功能完善的操作系统，来加深对操作系统中进程管理、内存管理、文件管理等概念及原理的理解，并真正了解一个操作系统是如何从无到有、一步步实现的。

1.2 开发环境

- **操作系统：**Ubuntu 64bits*(使用 VMware虚拟机)*
- **操作系统模拟器：**Bochs开源模拟器
- **代码编辑器：**Visual Studio Code
- **代码语言：**C语言、汇编语言

1.3 项目选题及达成指标

- **项目选题：**完成《Orange's：一个操作系统的实现》项目要求
- **达成指标：**
  - **B级：**本系统对文件系统进行重新实现，新增代码量达到相关模块代码的一半。

- **C级：**在参考源码上实现了系统级应用，如磁盘、工具台、进程管理等，通过调用较多系统API实现对系统的检测和控制。
- **D级：**在参考源码上实现了用户级应用，包括五个小游戏、日历、计算器。

2.设计方案

2.1 概述

本系统以《Orange's：一个操作系统的实现》一书中的源代码为基础，实现了一个简单的操作系统，并重新实现了其中文件系统的部分，包括多级目录，文件/目录的增加及删除，文件的读写等；同时通过调用部分系统API实现了数独、黑白棋、井字棋等小游戏，以及日历、计算器等工具，还为其添加了开机动画以及睡眠与唤醒功能。

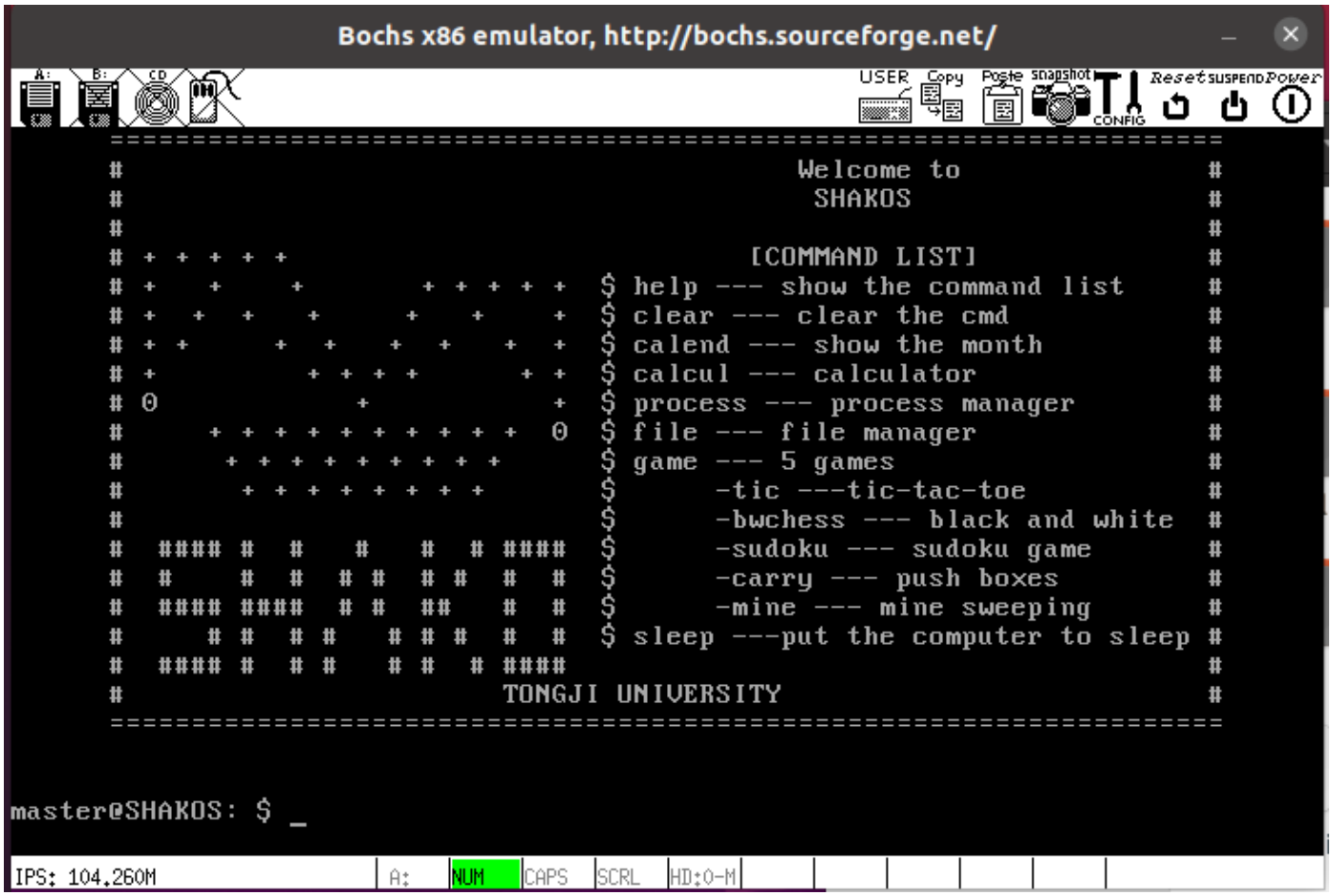
2.2 系统功能及使用说明

- 在 yeezyOS 打开终端
- 输入 bochs -> c（若对源文件有改动则在输入bochs之前输入make image）
- 进入 yeezyOS 操作系统

2.2.1 开机动画

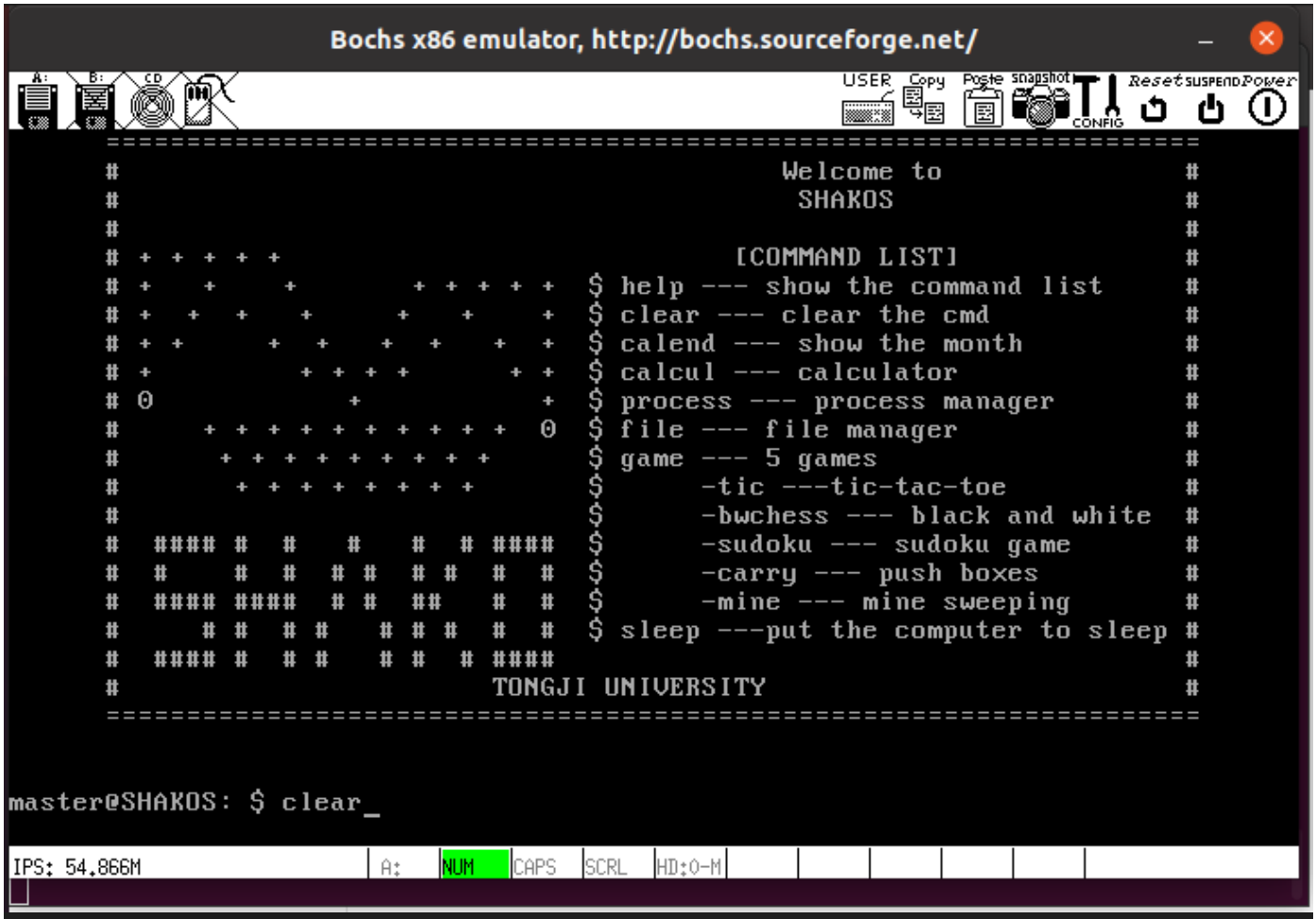
2.2.2 帮助界面/主界面

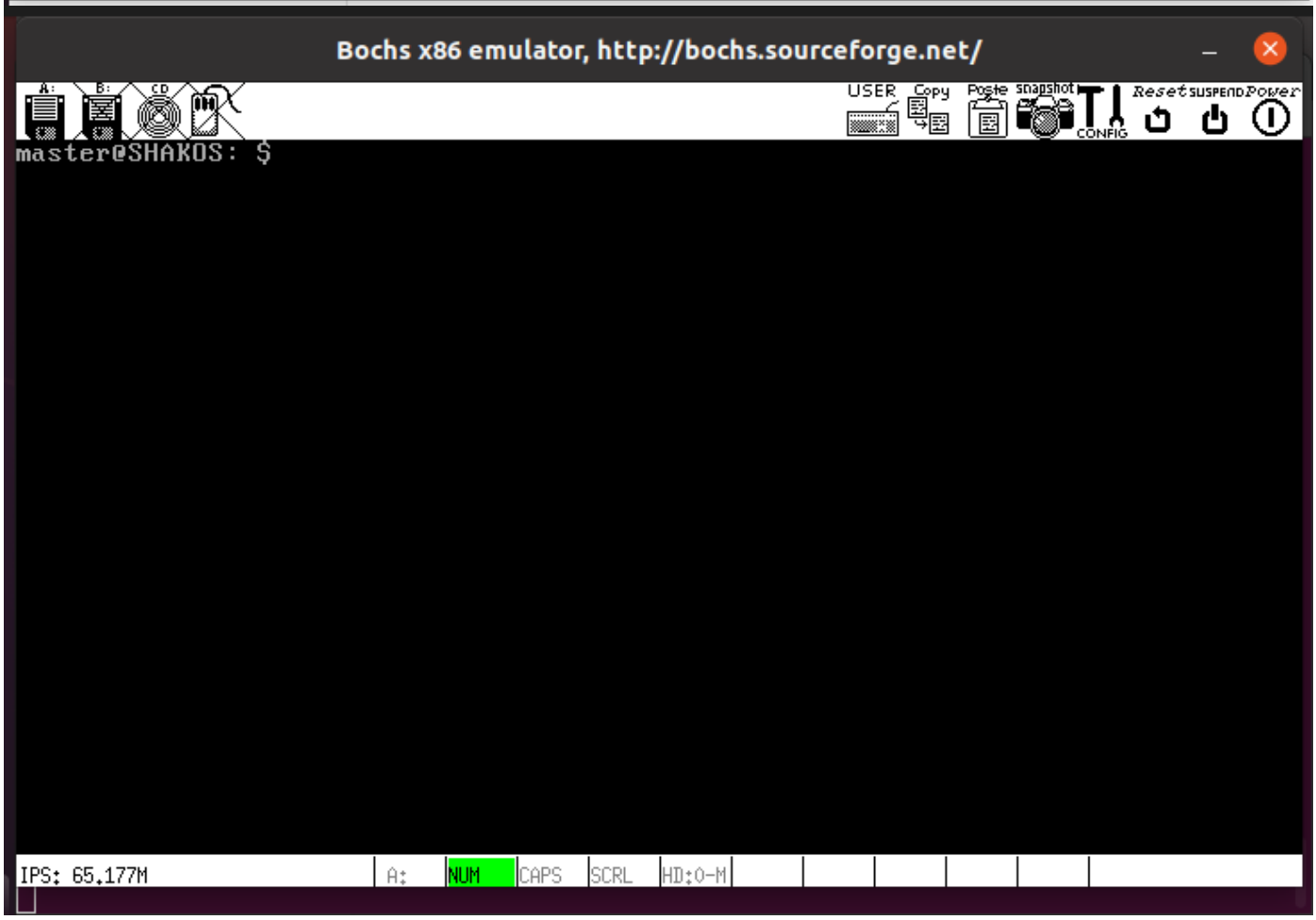
菜单显示系统提供的所有功能编号及概要。输入各命令或其编号，进入相应功能界面。 若想要回到 帮助界面/主界面 ， 在命令行中输入“help”按下回车即可 。



2.2.3 清屏功能

当需要清屏时，输入`clear`按下回车即可。

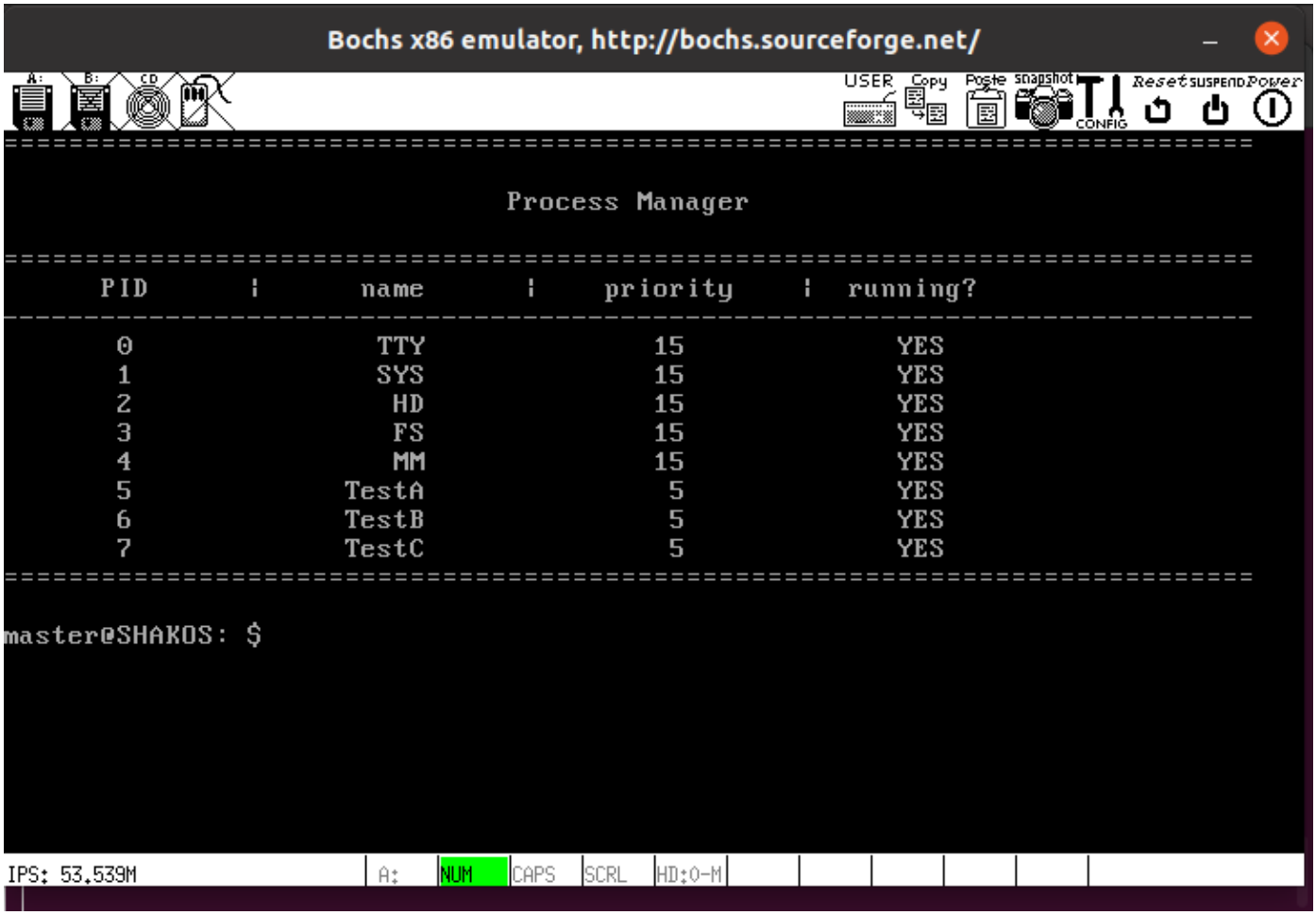

Bochs x86 emulator, http://bochs.sourceforge.net/
Welcome to
SHAKOS
[COMMAND LIST]
$ help --- show the command list
$ clear --- clear the cmd
$ calend --- show the month
$ calcul --- calculator
$ process --- process manager
$ file --- file manager
$ game --- 5 games
-tic ---tic-tac-toe
-bwchess --- black and white
-sudoku --- sudoku game
-carry --- push boxes
-mine --- mine sweeping
$ sleep ---put the computer to sleep
TONGJI UNIVERSITY
master@SHAKOS: $ clear
IPS: 54.866M


Bochs x86 emulator, http://bochs.sourceforge.net/
master@SHAKOS: $
IPS: 65.177M

2.2.4 进程管理

描述

用户输入指令**“process”**，进入进程管理功能，系统将打印出当前所有进程的PID、进程名称、优先级（15表示系统级进程，5表示用户级进程）以及运行状态。



功能实现

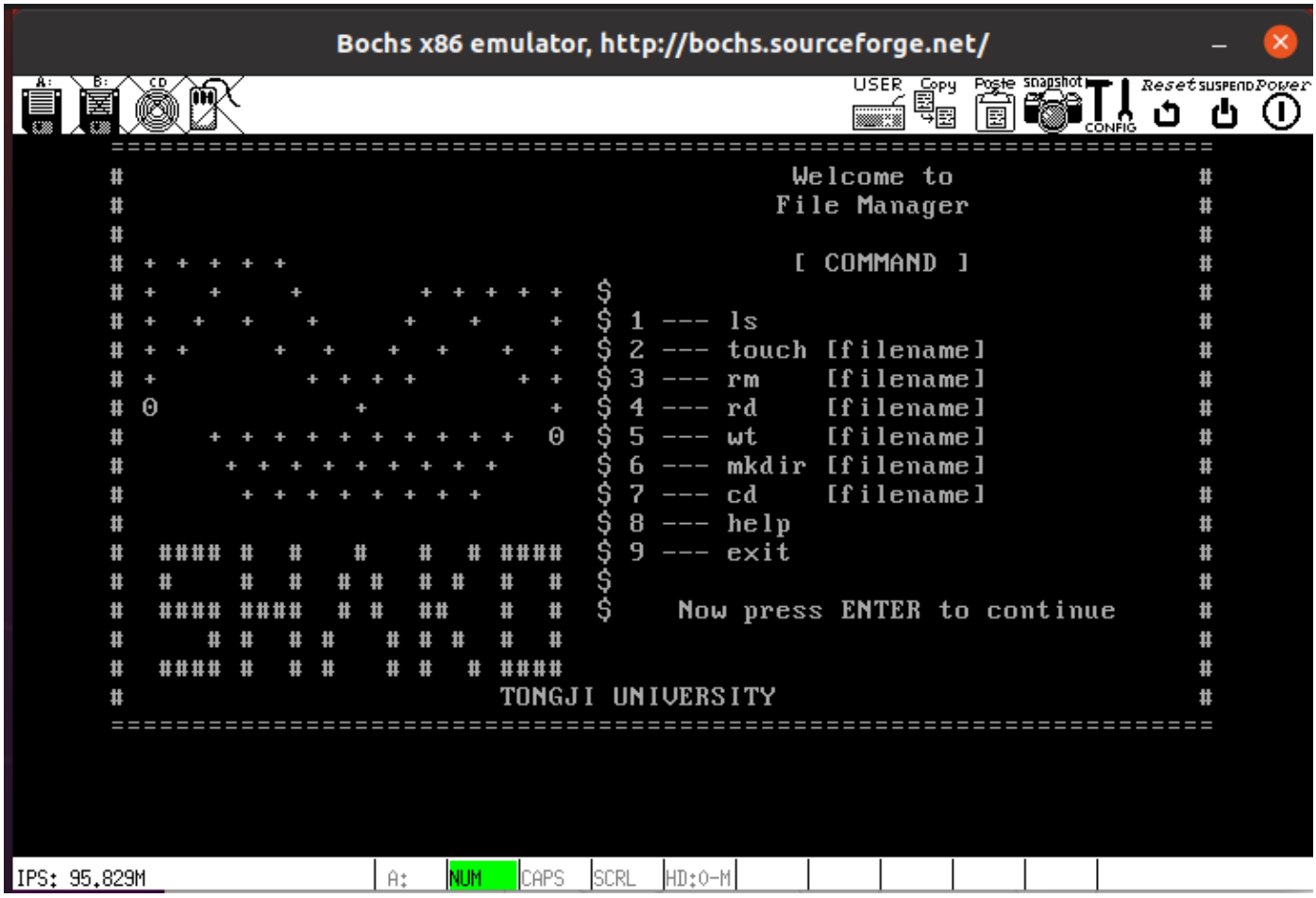
- 调用`process_manager()`，对进程列表proc_table[]进行遍历，逐个打印每个进程的信息，其中宏`NR_TASKS + NR_PROCS`表示所有进程的数量。

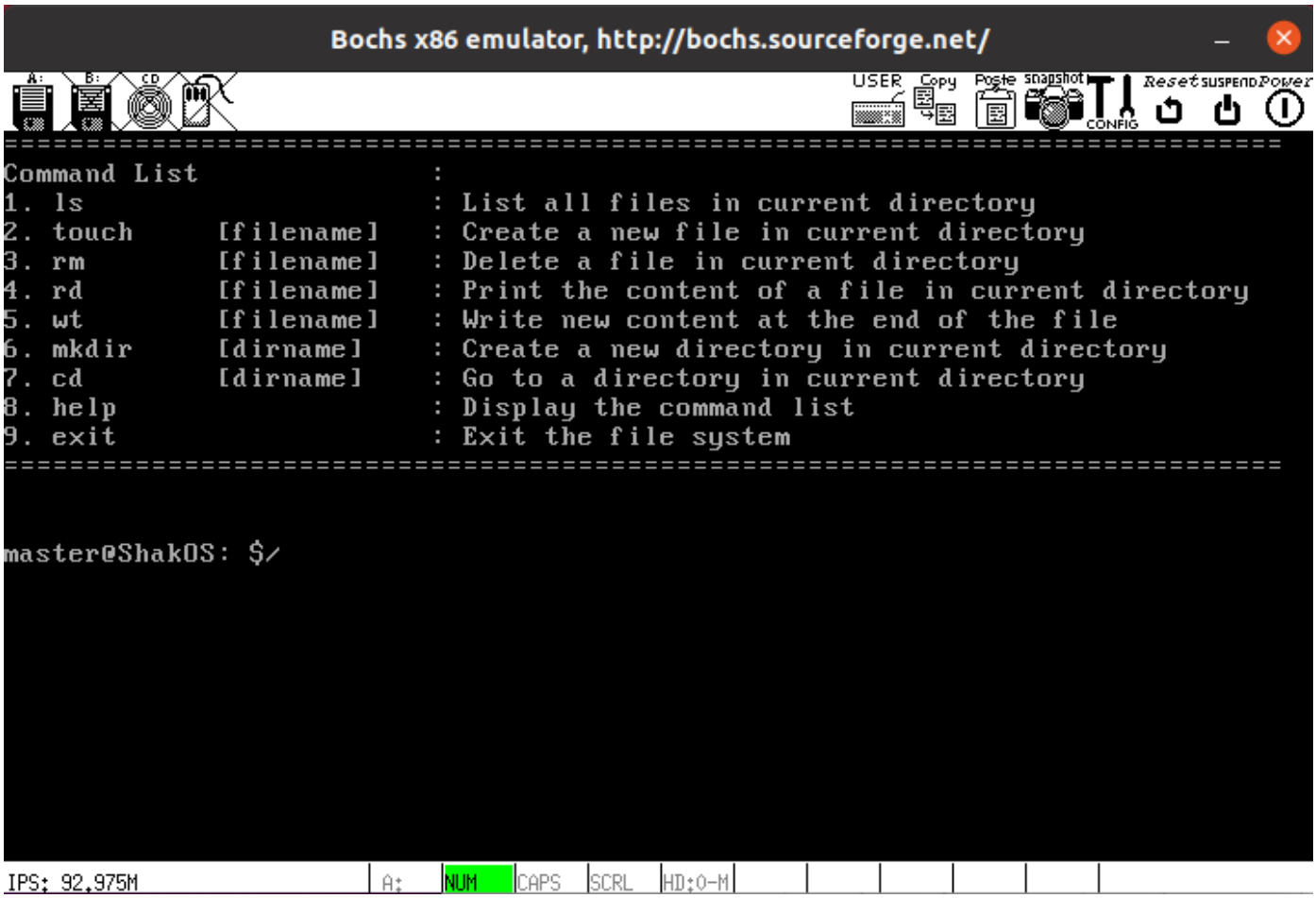
2.2.5 文件管理

2.2.5.1 概述

文件管理系统模拟Linux的命令行部分指令操作，设置初始用户为home，实现了可以长期保存的基于多级目录下的文件系统。主要的操作有：1、**ls指令**，查看当前目录下所有的目录及文件名；2、**touch指令**，创建文件存于当前目录下；3、**mkdir指令**，创建子目录存于当前目录下；4、**rm指令**删除文件；5、**rd指令**，读文件；6、**wt指令**，写文件；7、**cd指令**，进入多级目录。

用户通过在控制台中输入**“file”**指令进入文件管理系统帮助页面，随后摁下回车即可进入文件管理系统。

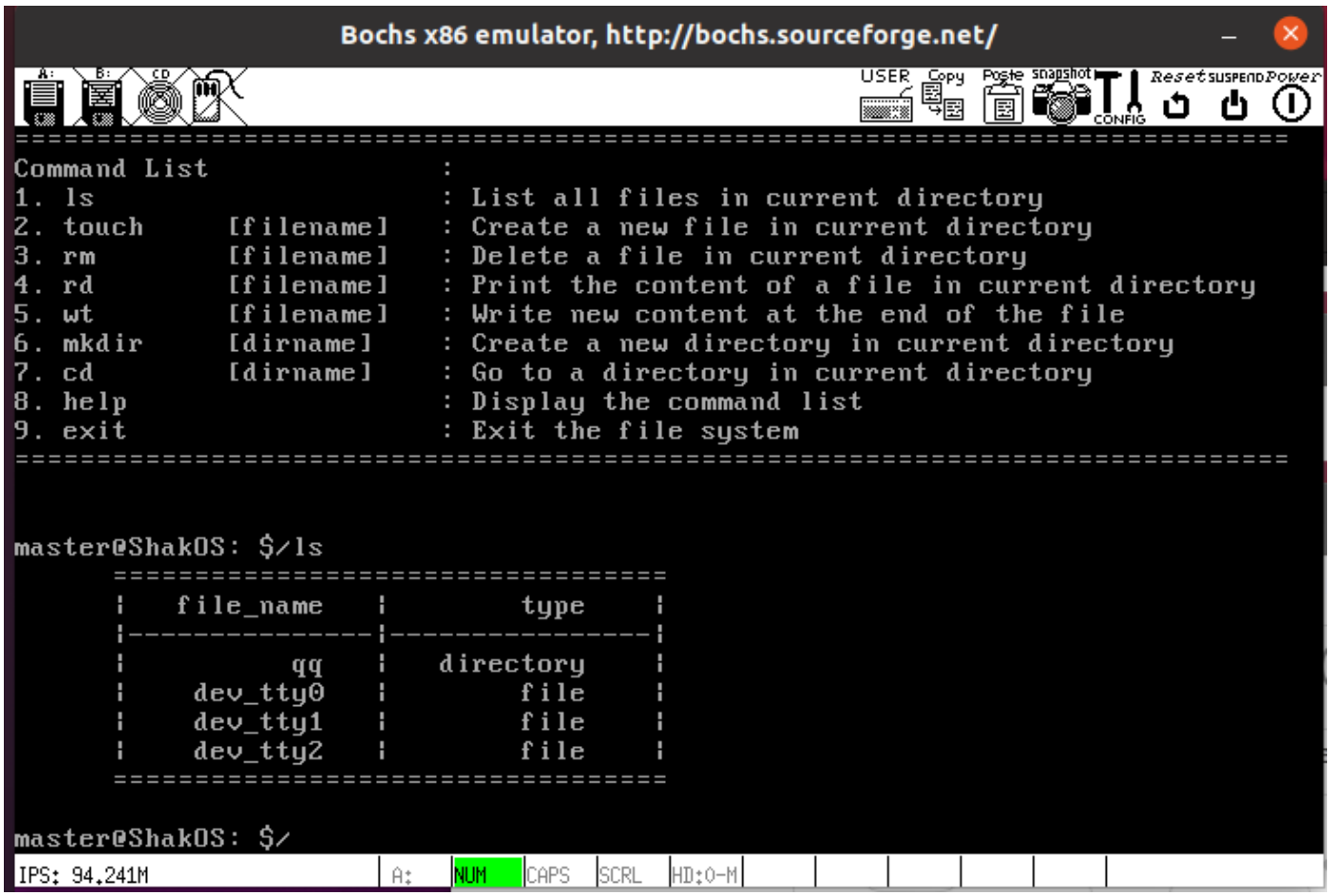

Bochs x86 emulator, http://bochs.sourceforge.net/
Welcome to
File Manager
[COMMAND]
$ 1 --- ls
$ 2 --- touch [filename]
$ 3 --- rm [filename]
$ 4 --- rd [filename]
$ 5 --- wt [filename]
$ 6 --- mkdir [filename]
$ 7 --- cd [filename]
$ 8 --- help
$ 9 --- exit
Now press ENTER to continue
TONGJI UNIVERSITY
IPS: 95.829M


Bochs x86 emulator, http://bochs.sourceforge.net/
Command List :
1. ls : List all files in current directory
2. touch [filename] : Create a new file in current directory
3. rm [filename] : Delete a file in current directory
4. rd [filename] : Print the content of a file in current directory
5. wt [filename] : Write new content at the end of the file
6. mkdir [dirname] : Create a new directory in current directory
7. cd [dirname] : Go to a directory in current directory
8. help : Display the command list
9. exit : Exit the file system
master@ShakOS: $/
IPS: 92.975M

2.2.5.2 ls指令

2.2.5.2.1 实现效果

用户进入文件系统后，输入指令**“ls”**，系统打印出当前文件目录下的全部文件及目录。



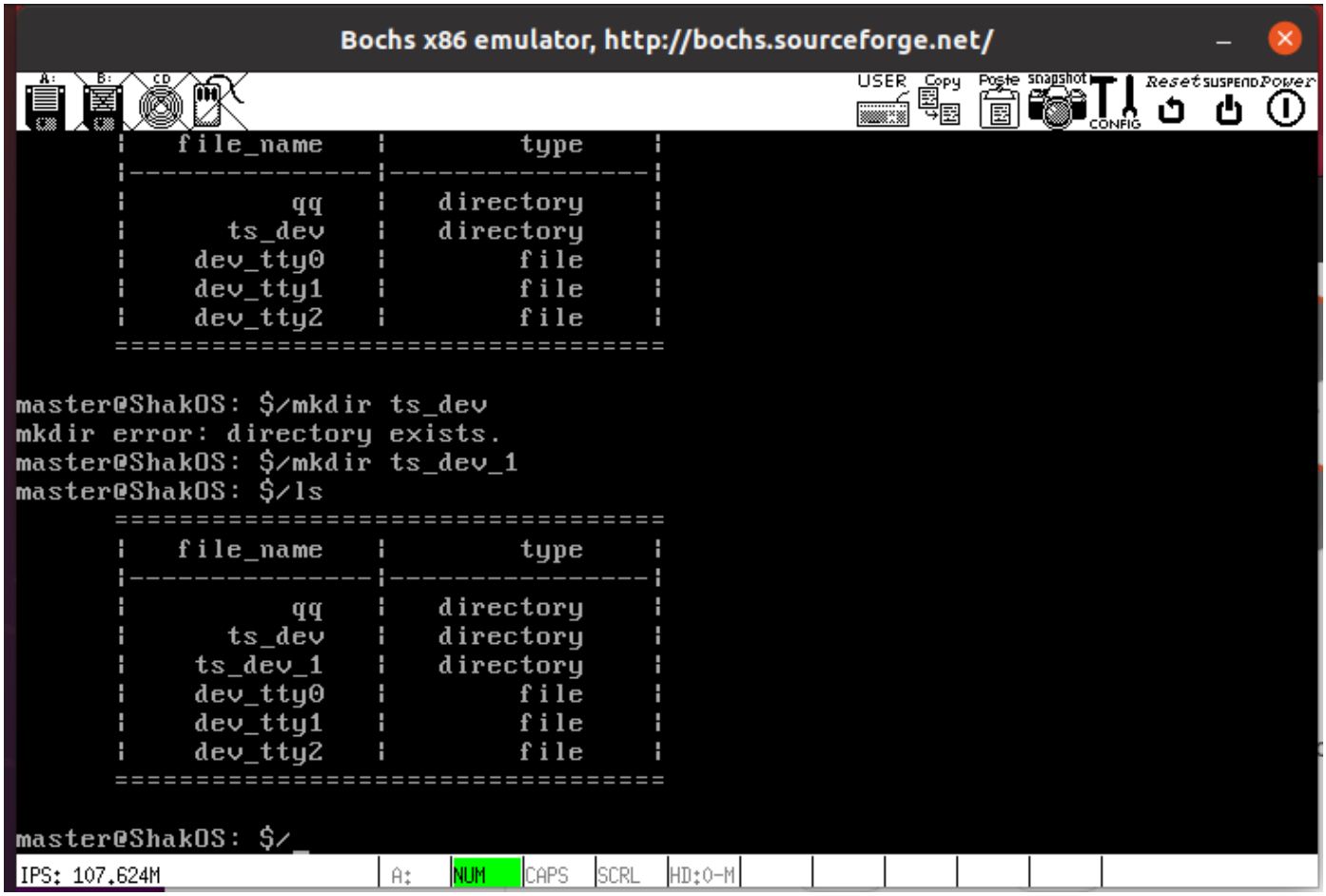
2.2.5.2.2 实现方法

调用系统函数ls()方法，找到当前目录下的全部文件名，并展示当前目录下的所有文件及目录。

2.2.5.3 mkdir指令

2.2.5.3.1 实现效果

用户进入文件系统后，输入指令**“mkdir + dirname”**，系统即在当前目录下创建一个新的目录文件并命名为dirlsname。若创建成功，系统提示“DIR created”的消息，若创建失败则系统提示“Failed to create a new directory”的消息。



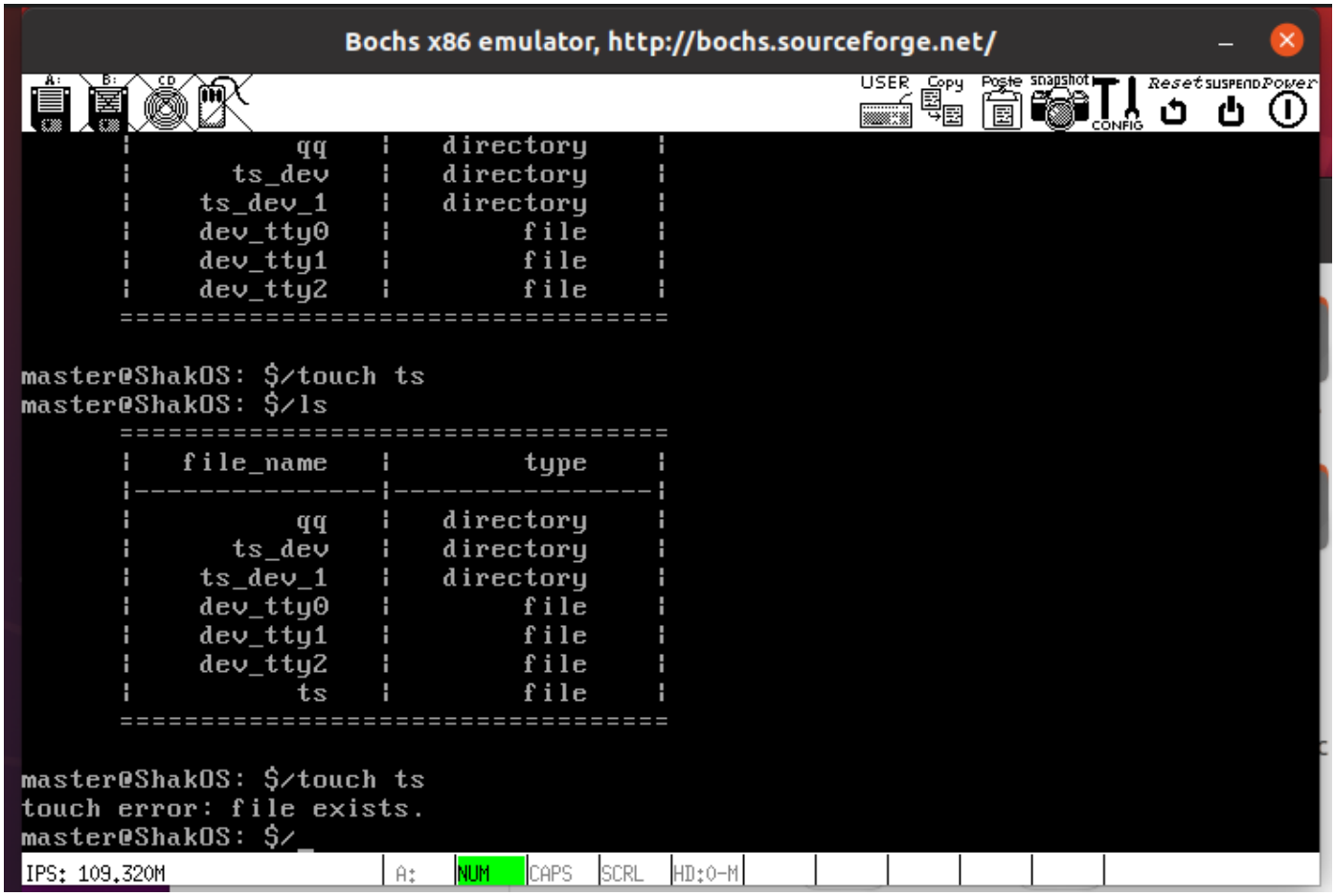
2.2.5.3.2 实现方法

根据传入的目录参数，通过调用open()方法创建对应类型的目录，根据方法返回值判断是否创建成功并给用户发出反馈消息。

2.2.5.4 touch指令

2.2.5.4.1 实现效果

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



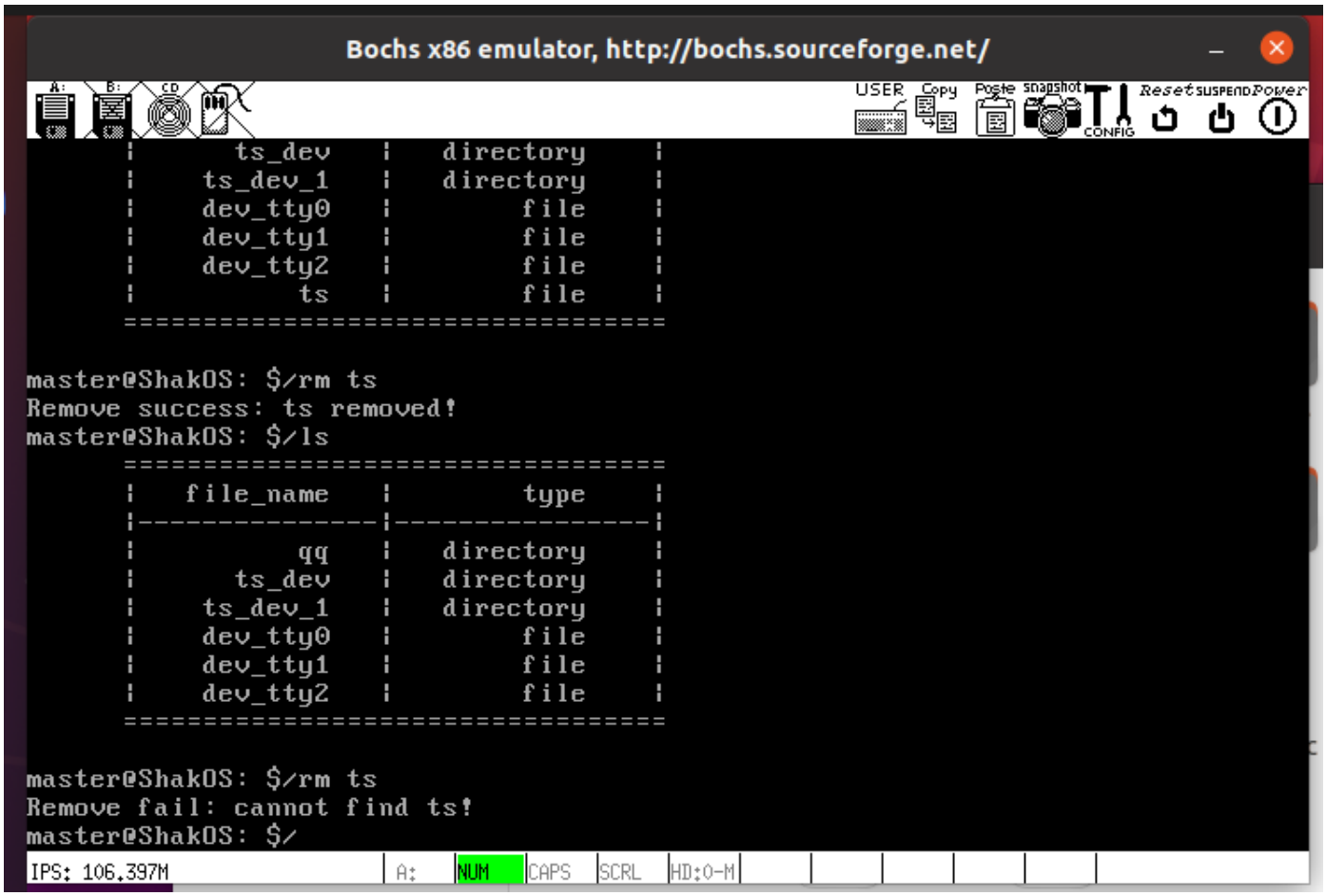
2.2.5.4.2 实现方法

根据传入的type类型（I_REGULAR），通过调用open()方法创建对应类型的新文件，根据方法返回值判 断是否创建成功并给用户发出反馈消息 。

2.2.5.5 rm指令

2.2.5.5.1 实现效果

用户进入文件系统某一目录后，可以输入“rm”在当前目录下进行文件删除操作。若删除普通文件，则 输入操作“rm filename”，系统执行删除操作并返回是否删除成功标志；若删除目录文件，则输入操作 “rm -r dir_name”，系统执行删除目录操作并返回是否成功删除标志。



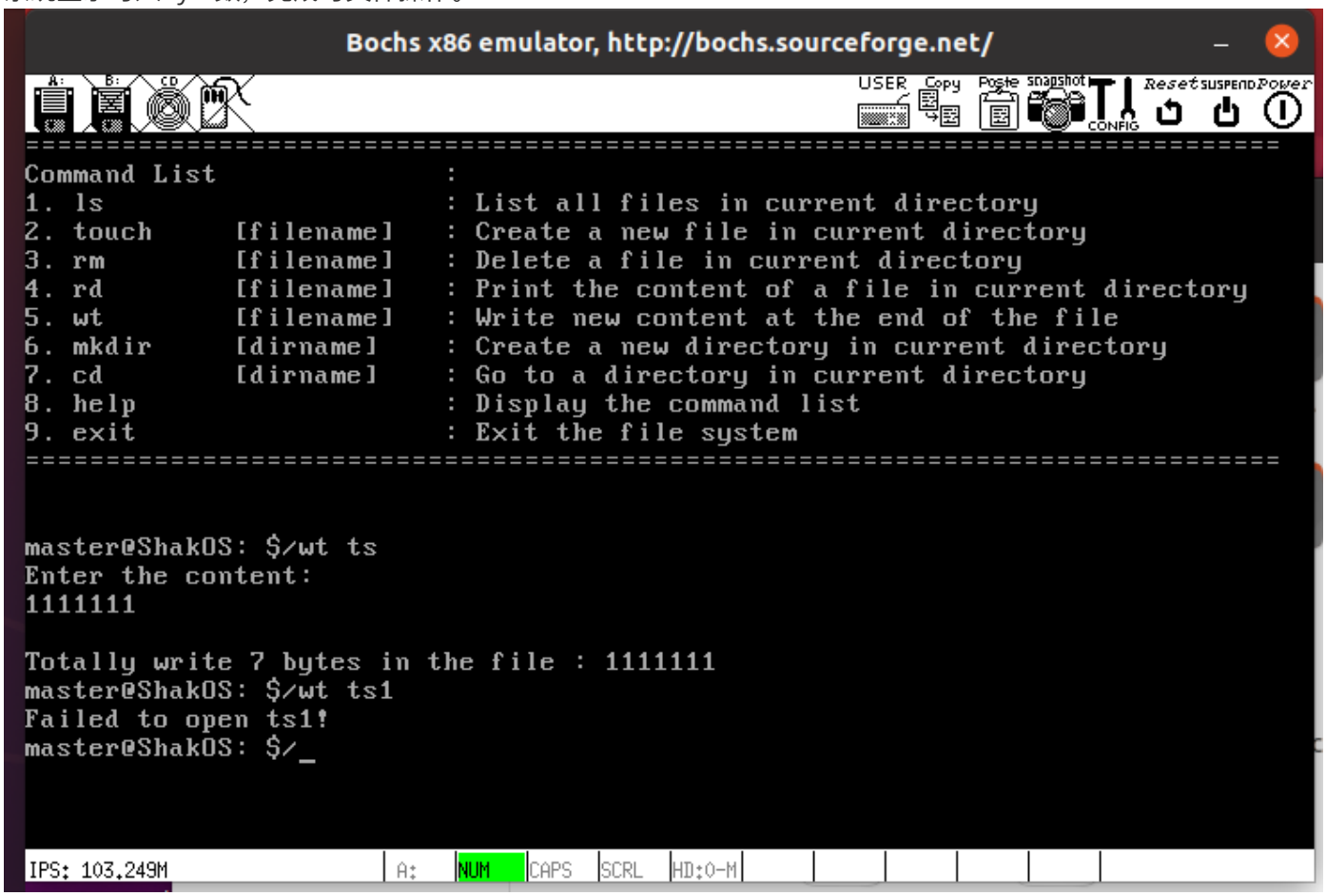
2.2.5.5.2 实现方法

通过调用unlink()API实现文件删除，通过递归调用dir_unlink()实现目录文件的级联删除。

2.2.5.6 wt指令

2.2.5.6.1 实现效果

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



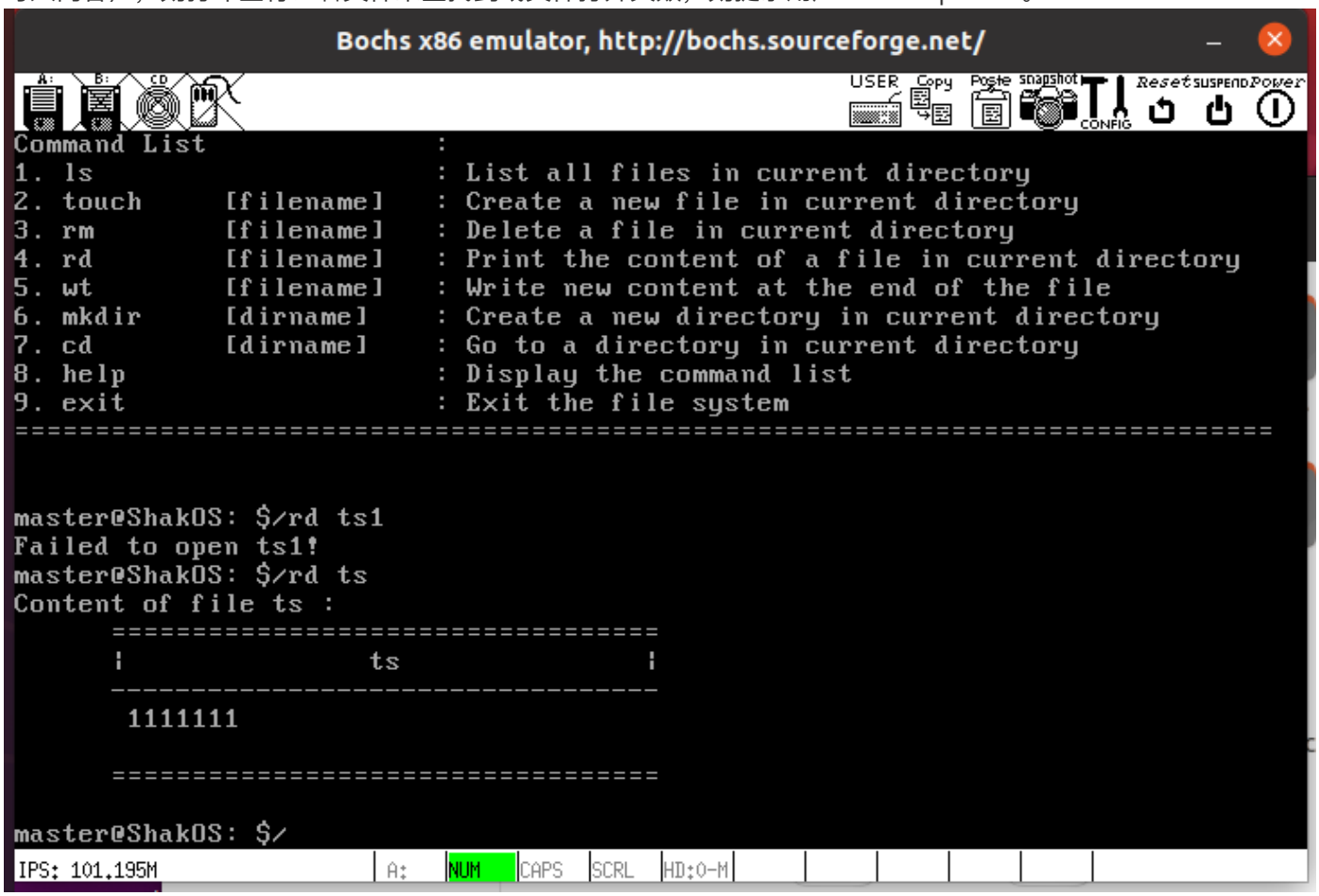
2.2.5.6.2 实现方法

调用open()方法定位文件，然后调用write()方法对文件进行写操作。

2.2.5.7 rd指令

2.2.5.7.1 实现效果

用户输入“rd+filename”指令，查看该文件内容，若文件已写入内容，则显示文件内容；若文件已创建 但为空（未写入内容），则打印空行；若文件未查找到或文件打开失败，则提示用户“fail to open！”。



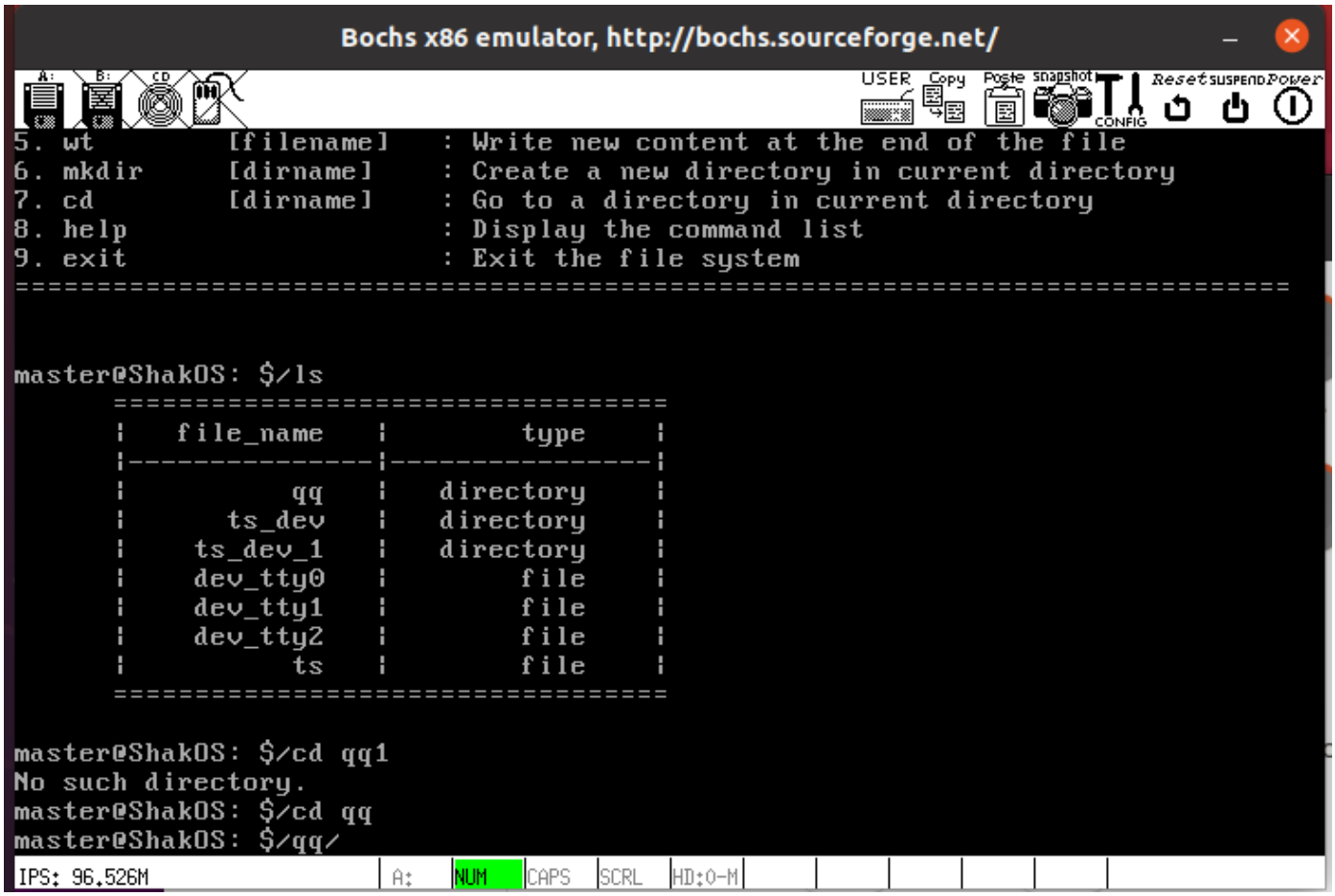
2.2.5.7.2 实现方法

调用open()方法定位文件，然后调用read()方法对文件进行读操作 。

2.2.5.8 cd指令

2.2.5.8.1 实现效果

用户输入“cd+directory”指令，系统定位到该相对路径所在的目录并检索其是否存在，如果存在则将 cur_dir变量后追加用户所输入的相对路径，在此后进行操作时，控制台屏幕上显示的当前目录为cd后进 入的目录，以后操作中涉及的目录均以此为基础进行定位。



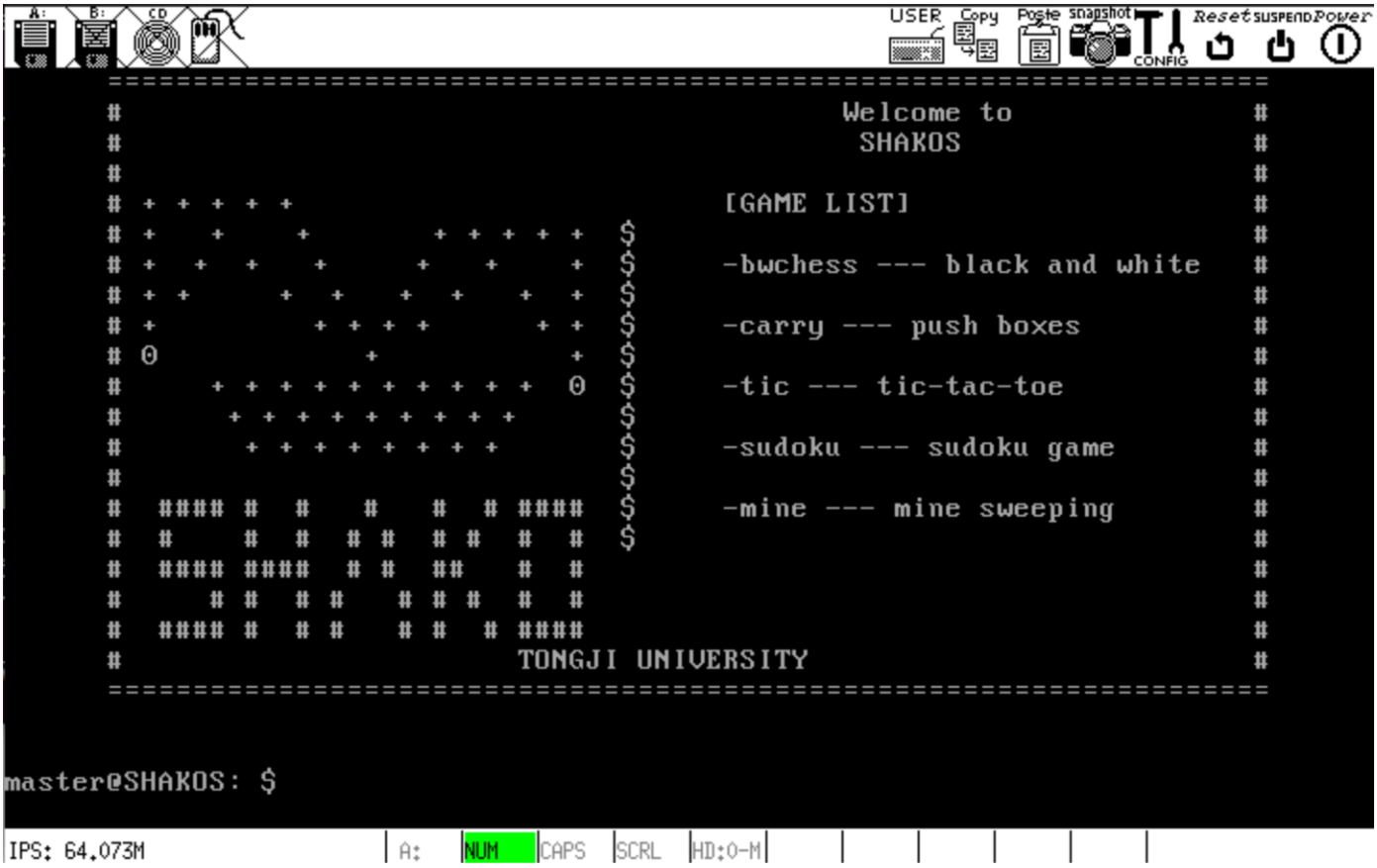
2.2.5.8.2 实现方法

调用open()方法确认要进入的目录是否存在，如果存在则有strcat()函数更新cur_dir。

2.2.6 游戏及工具

2.2.6.1 游戏界面

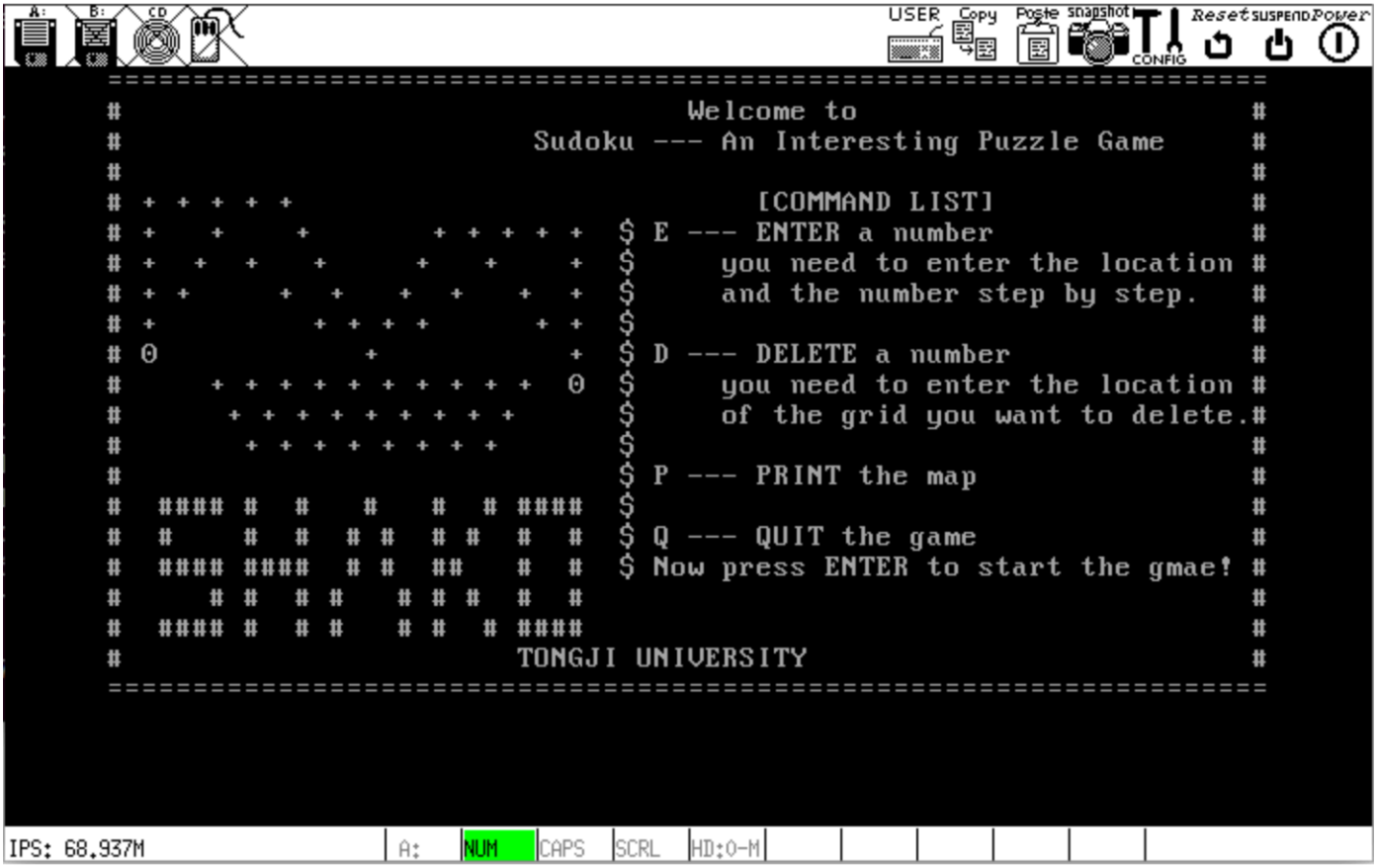
用户输入指令**"game"**，进入游戏界面，控制台将显示所有游戏简介及相应指令。



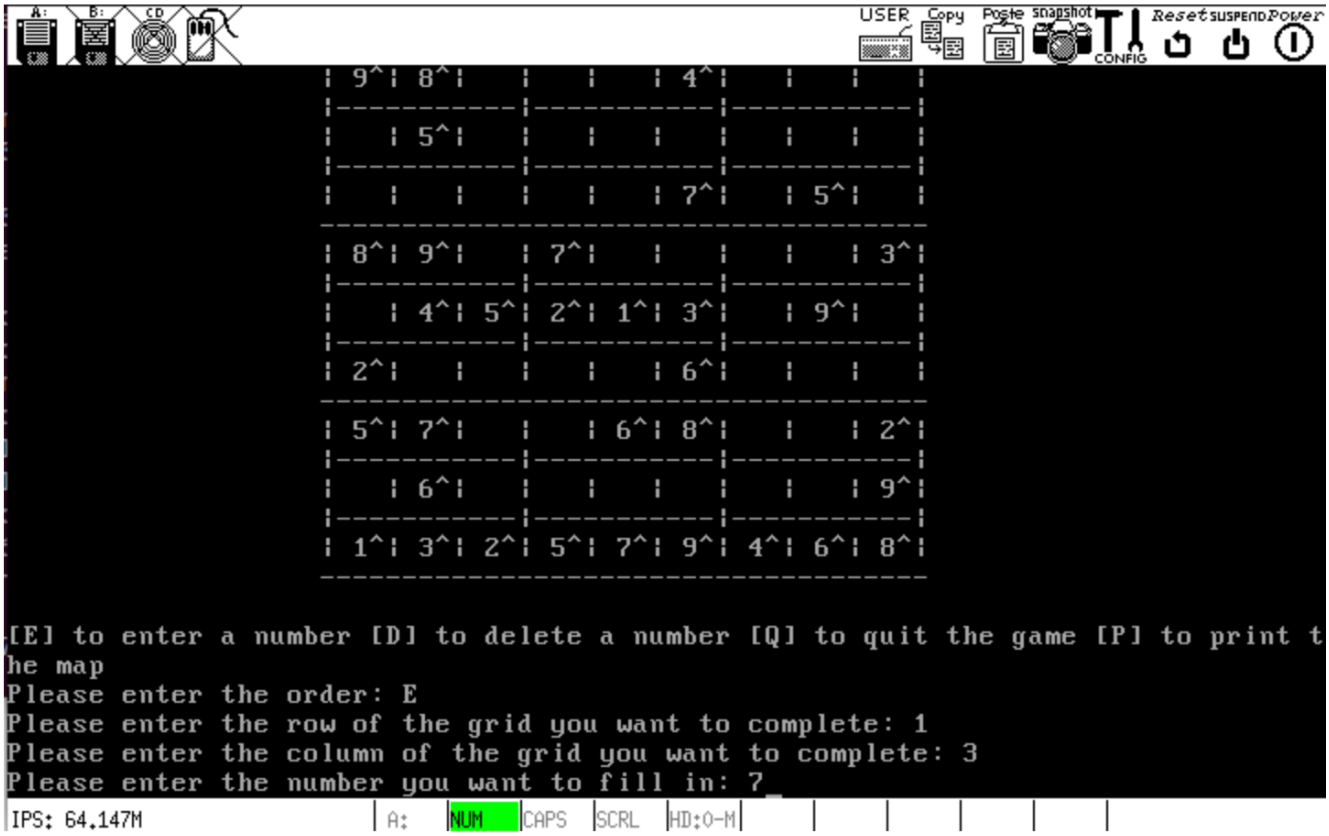
2.2.6.2 sudoku（数独）

操作描述

- 用户输入指令**“sudoku”**，进入数独游戏界面，控制台将显示游戏的操作说明，用户按下回车键即开始游戏



- 用户可以通过指令在数独棋盘中输入或删除数字，每次操作后将打印新的棋盘



功能实现

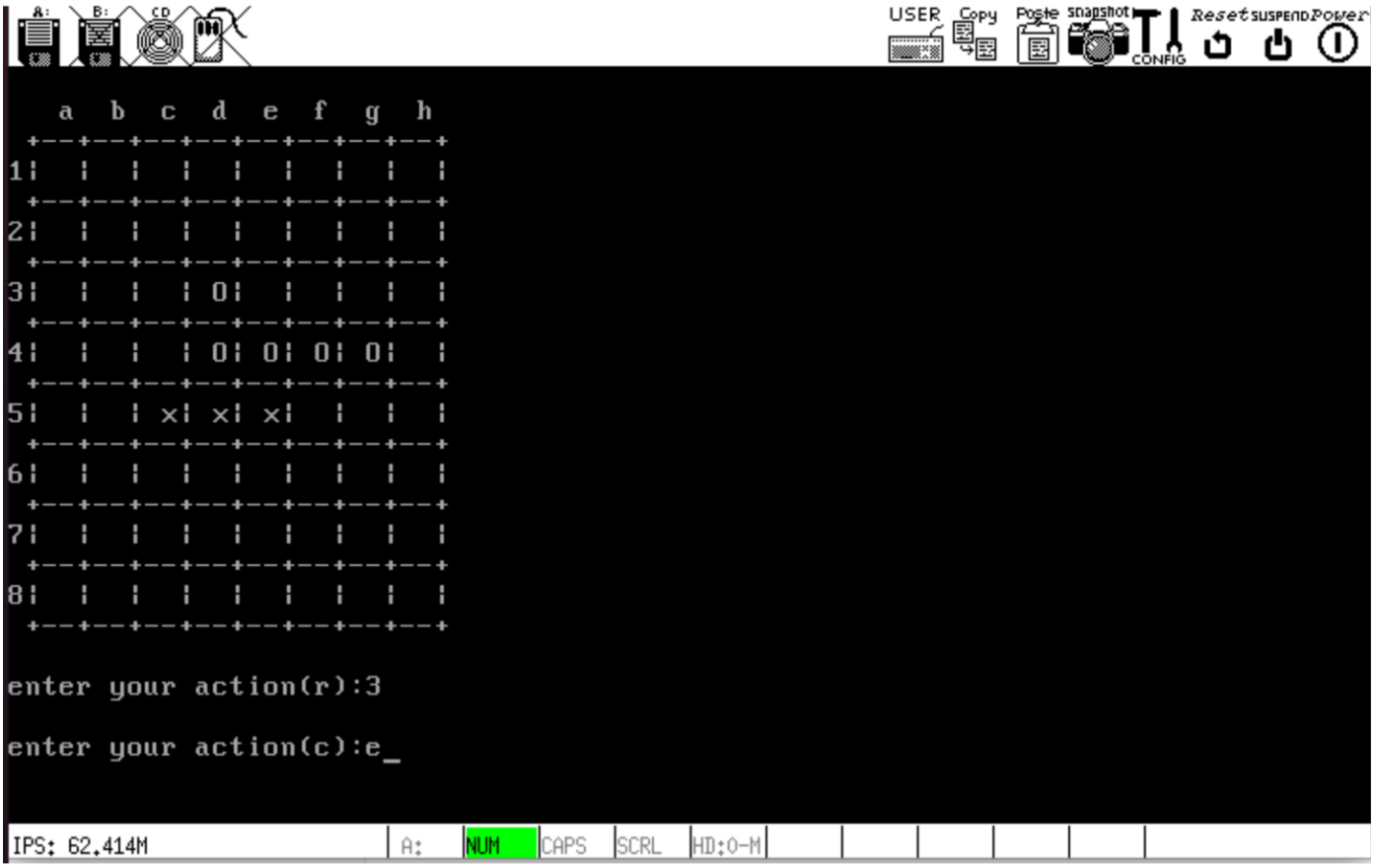
- 采用深度优先搜索，dfs(int i, int j, bool *success)，来建立初始的棋盘，不断向下递归判断何处放置初始数字合理，直到最后返回success。调用getmap()随机生成一个数独棋盘，并随机取一定数量单元格作为初始给定的单元格
- 调用sudoku_main()接收用户的指令进行游戏
- is_legal(int i, int j)用来保证在建立棋盘时的合理性，九宫格、同一行、同一列内不能有相同的数字。
- 每次用户完成操作后，调用 printmap() 打印新棋盘。
- 若当前已被填满，则调用 check_win() 判断用户是否获胜。

2.2.6.3 bwchess（黑白棋）

操作描述

- 用户输入指令**“bwchess”**，进入数独游戏界面，控制台将显示游戏的操作说明，用户按下回车键即开始游戏

- 系统默认玩家持黑子，AI持白子，玩家可以选择输入1或0来选择先手或后手
- 用户可以通过指令输入将要落子的行列进行落子，每次操作后将打印新的棋盘



- 当棋盘下满或双方无法钧无法落子时，系统将判定输赢并计算分数，显示游戏结算信息

功能实现

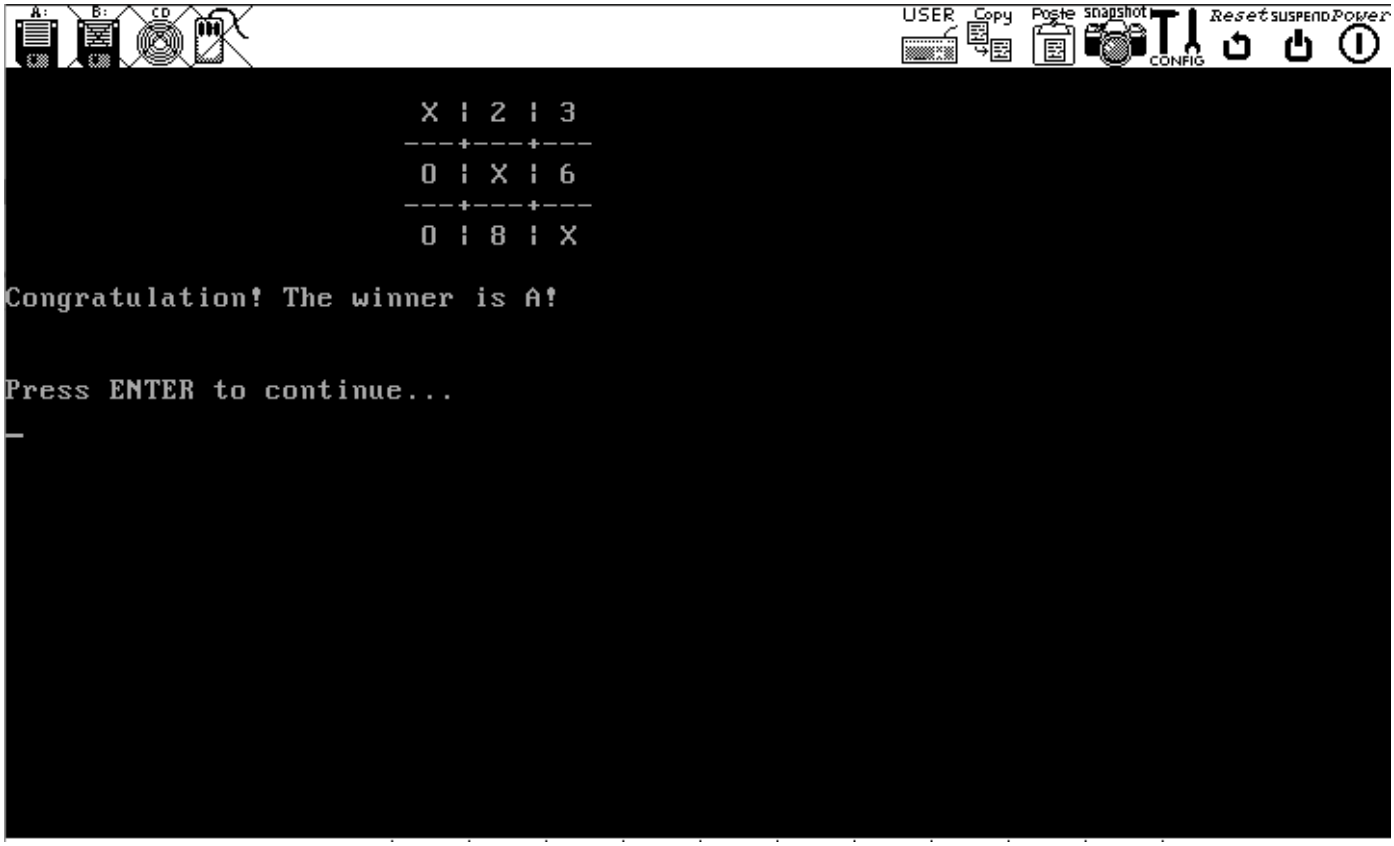
- 调用Exa()函数计算可供落子的位置的个数，用于判断一方是否还有位置落子
- 调用Calsore()函数通过计算最后棋子的个数来计算玩家最后的得分
- 调用show()函数来每次打印棋盘
- 调用Foeplay()函数来让计算机思考下棋的位置，其中调用Hint()函数，将遍历每个能落子的地方所得到的当前分数，最后Hint()将返回一个最高得分的情况，Foeplay()才决定机器最后的落子位置
- 调用bwchess_main()函数来接受用户指令进行游戏

2.2.6.4 tic-tac-toe(井字棋)

操作描述

- 用户输入指令**“tic”**，进入井字棋游戏界面，控制台将显示井字棋的操作说明，用户按下回车键即开始游戏

- 游戏支持两位玩家进行对战，两位玩家依次输入数字1-9（表示落子的位置）进行落子（若输入0则可退出游戏小程序）当输入位置已有棋子，系统会提醒玩家重新输入。若其中一位玩家的三颗棋子连成一线，则该玩家获胜；若棋盘已满而无人获胜，则平局，将在屏幕显示最终游戏结果



功能实现

- 游戏代码写在user文件中的app.c中，再main.c中调用函数ticTacToe_main() 输出井字棋棋盘，接收用户输入的指令执行相应操作，并判断游戏最终结果

2.2.6.5 carrycraft(推箱子)

操作描述

- 用户输入指令**“carry”**，进入推箱子游戏界面，控制台将显示游戏的操作说明，用户按下回车键即开始游戏

- 玩家通过wasd输入来实现控制人物进行上下左右的行走,地图中w代表墙，P代表玩家，c代表箱子，T代表目的地。

- 状态图根据玩家的指令修改相应信息，每次操作后将打印新的地图,当所有的箱子都放置到目标地点后或者任务无法移动，系统将判定输赢，显示游戏结算信息。

功能实现

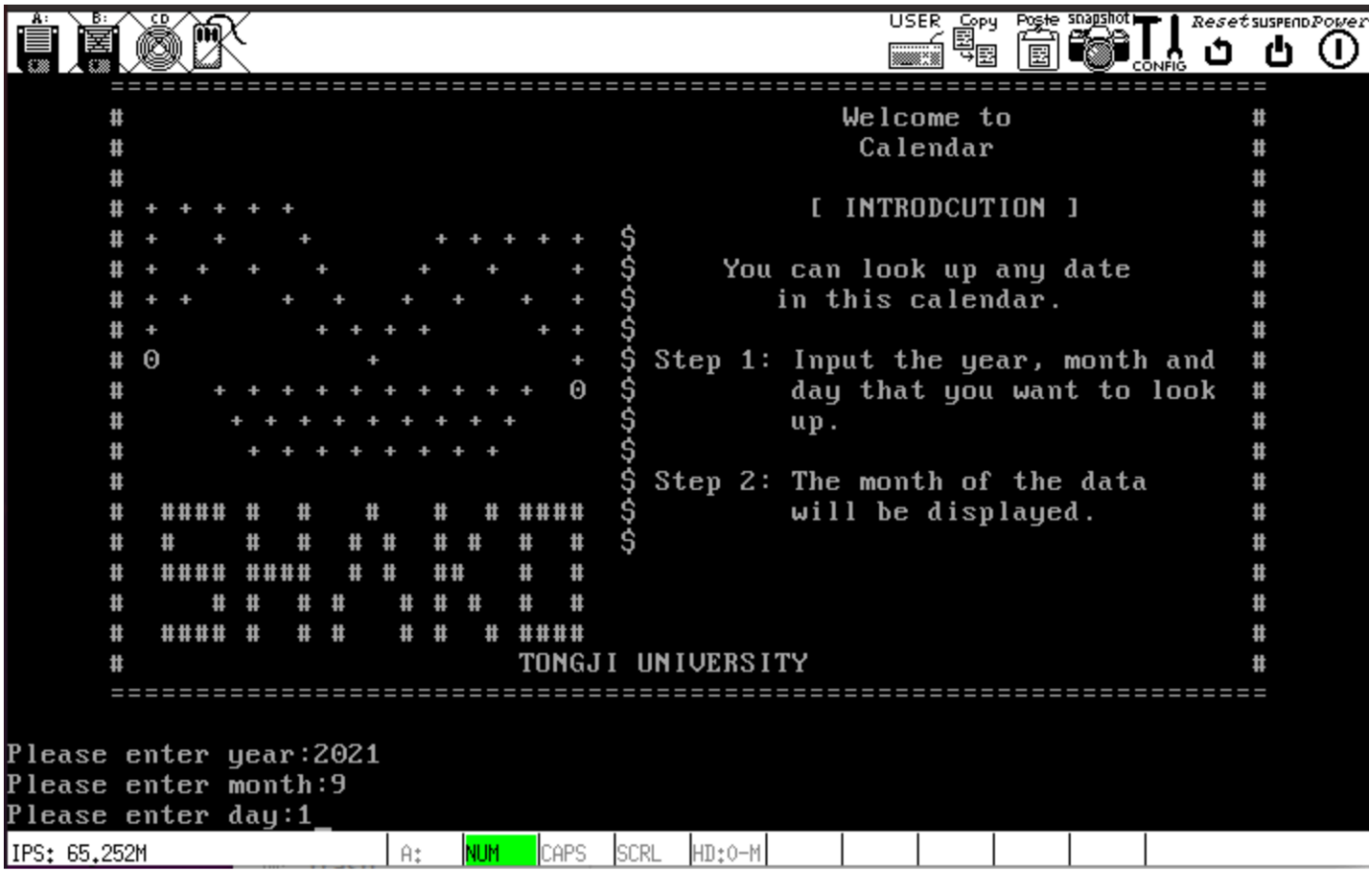
- 游戏代码写入app.c文件中，在main.c文件中调用接口carry_main()来接受用户指令进行游戏。

2.2.6.6 calendar（日历）

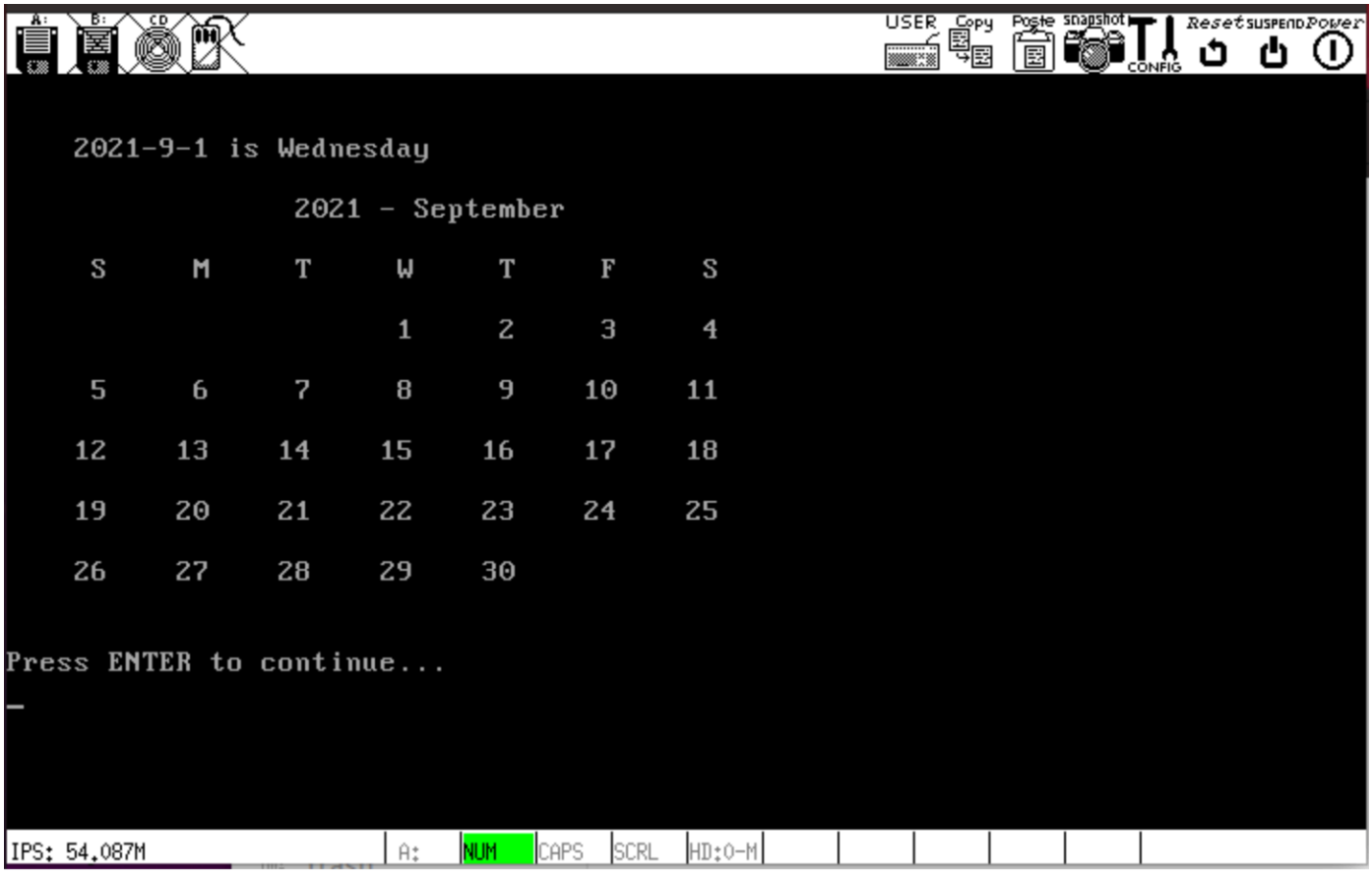
操作描述

- 用户输入指令**“calend”**，进入日历应用界面，控制台将显示日历的操作说明

- 用户输入想要查询的具体日期（年月日）



- 系统输出查询的具体日期是星期几，并将该月份的日历打印出来



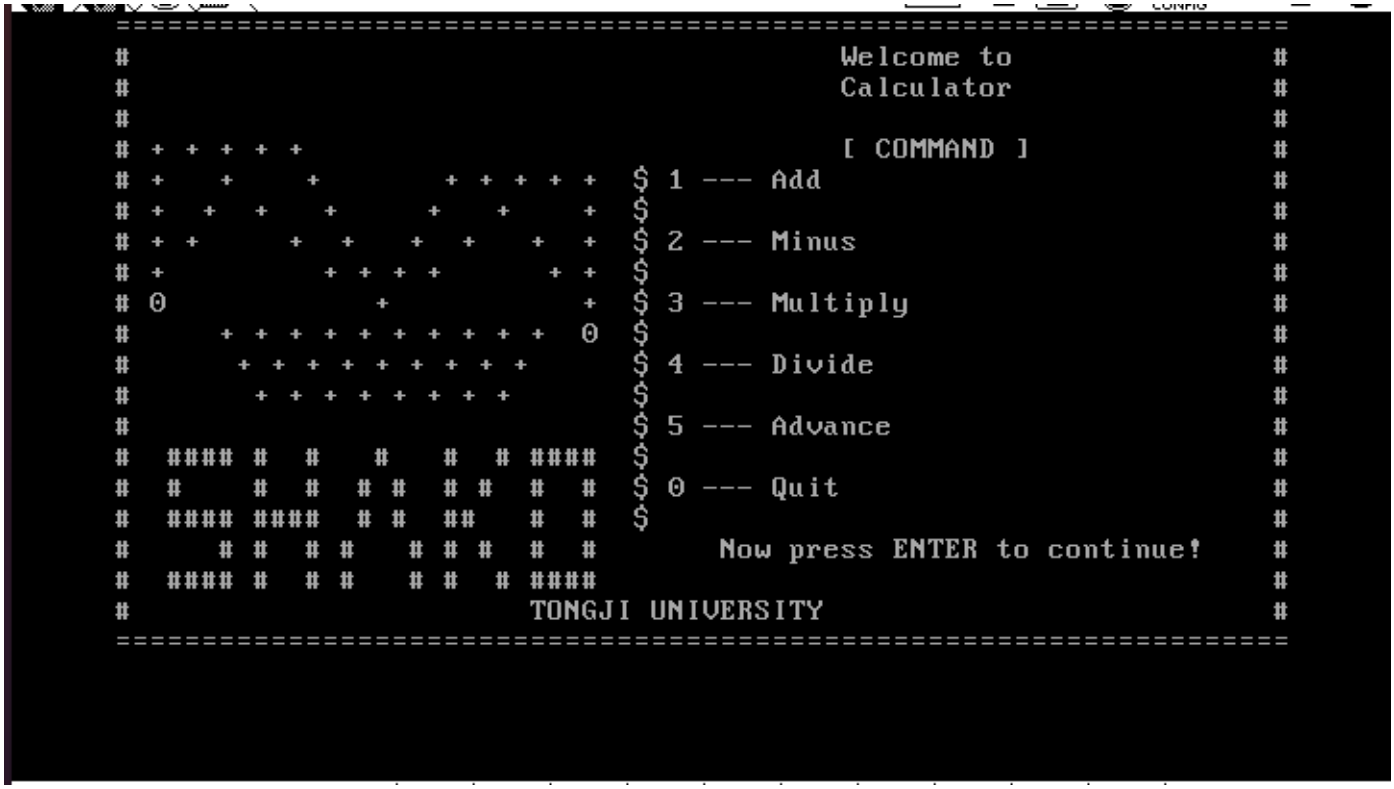
功能实现

- 调用weekday(int year, int month, int day)函数来判断输入日期是星期几
  - (year - 1) + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400 + total_day(year, month, day)中，(year - 1) / 4表示加上被4整除的年份，(year - 1) / 100表示需要去掉被100整除的年份，(year - 1) / 400表示需要加上被400整除的年份，之所以要year-1，是因为需要去掉输入的这一年。
- 调用display_month(int year, int month, int day)来打印该月的日历
  - 假设输入的日期是每月的第一天，将第一天是星期几计算出来后，按一周七天的顺序按序输出该月的每一天。
- 调用calendar_main(int *fd_stdin)函数来接受用户指令进行查询

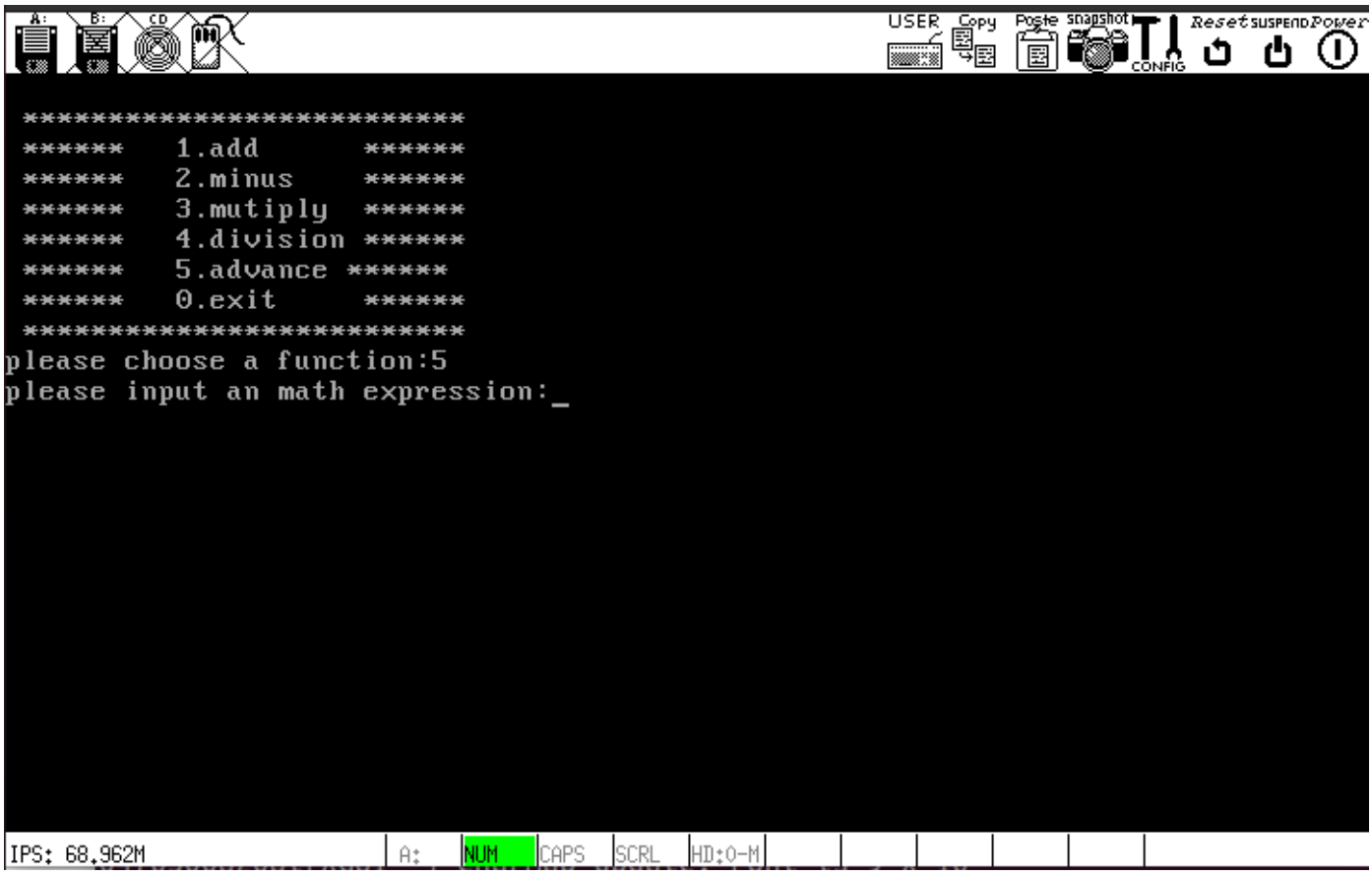
2.2.6.7 calculator（计算器）

操作描述

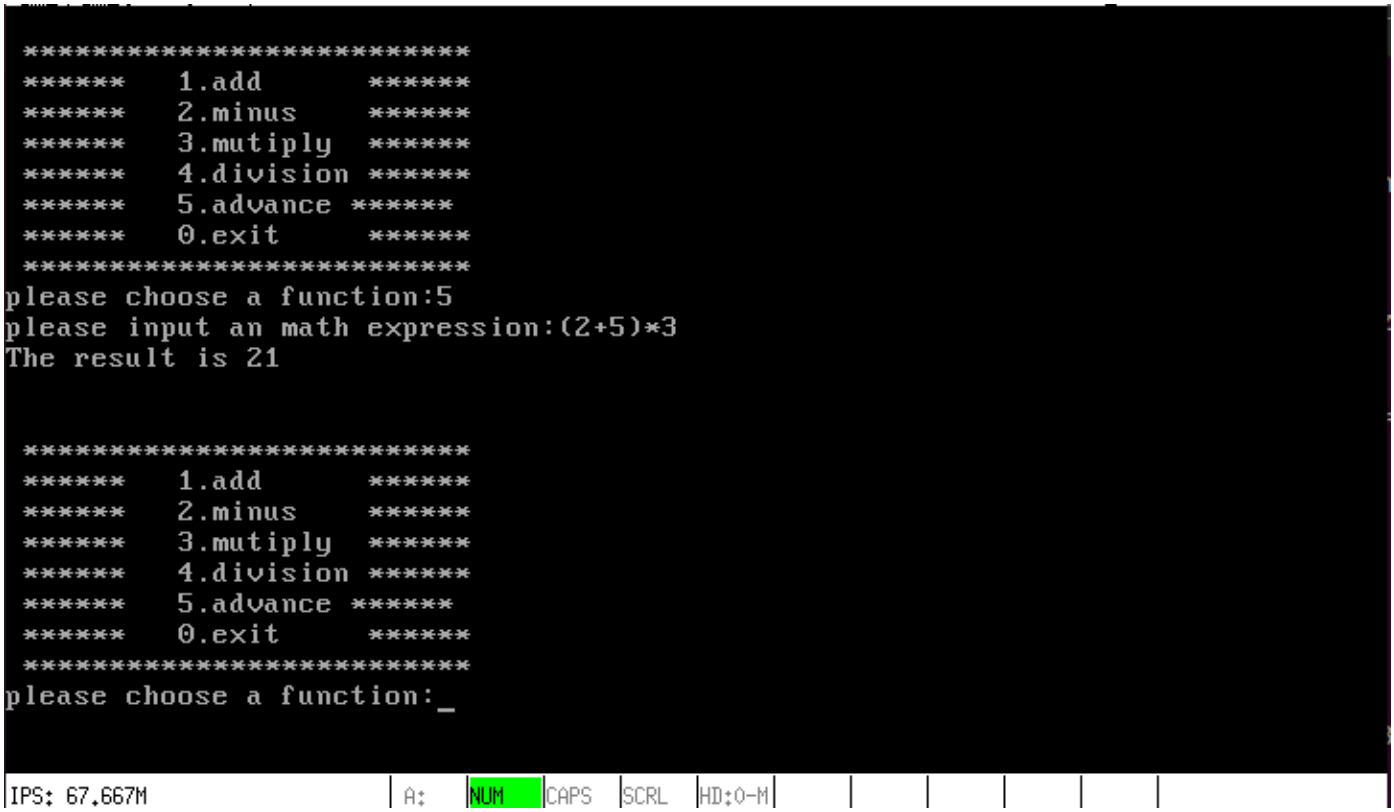
- 用户输入指令**“calcul”**，进入计算器应用界面，控制台将显示计算器的操作说明



- 输入要进行的操作



- 输入要计算的表达式或数字



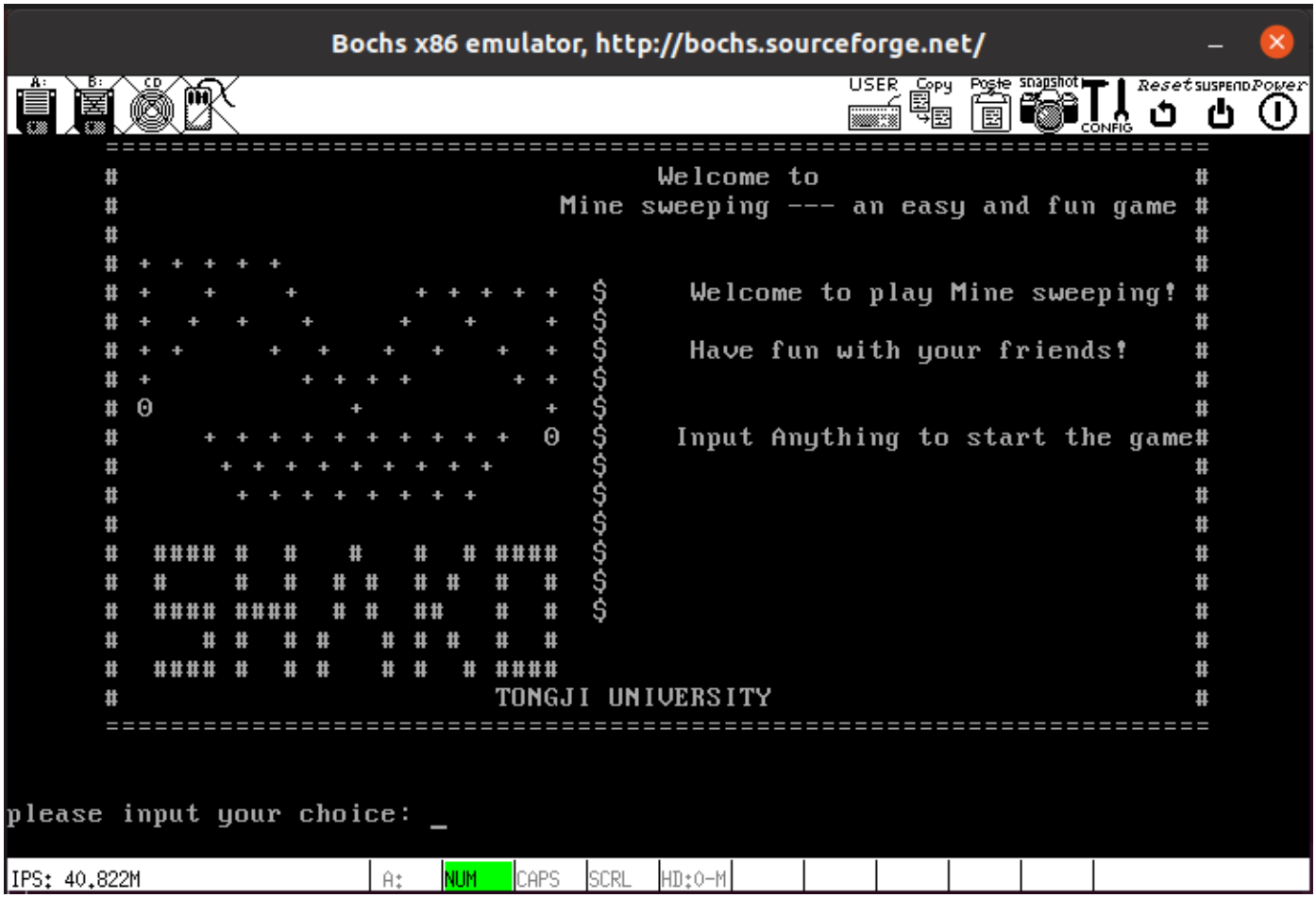
功能实现

功能代码写入app.c中，在main.c中调用calculator_main(&fd_stdin)函数使用计算器。

大致步骤如下：

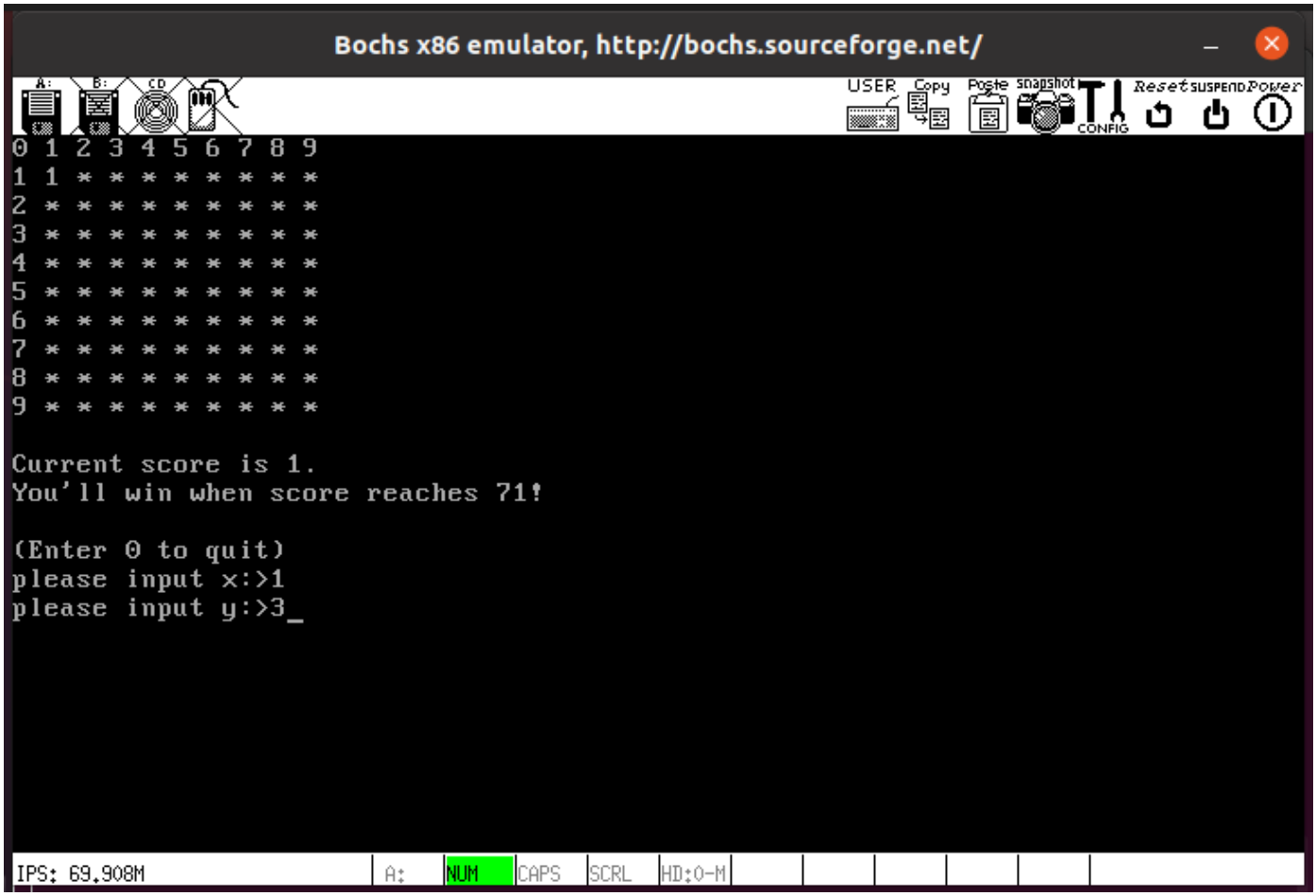
- 调用get_option(int *fd_stdin)函数让用户选择要进行的操作
- 用户输入要计算的数字
- 调用print_result()函数打印计算结果

2.2.6.8 mine sweeping(扫雷)



操作描述

- 用户输入**“mine”**进入界面，然后输入任意键即可开始游戏



- 用户可以通过指令在扫雷棋盘中翻开指定块，每次操作后将打印新的棋盘
- 用户成功排除所有雷（10个）时，将显示游戏获胜信息
- 当用户踩到雷时，系统判定结束自动退出

功能实现

- 调用InitBoard()生成两个棋盘，一个用于显示给用户，一个用于记录哪里埋雷
- 调用mine_main()接收用户的指令进行游戏

2.2.7 休眠功能

操作描述

- 用户输入指令**“sleep”**，即可进入休眠功能，系统将会锁住，必须按照提示输入相应的字符串，才可以解锁，要输入的字符串每次都会重新随机生成

功能实现

- 调用`sleep()`函数，生成随机字符串
- 接收用户输入信息并与字符串比较判断匹配与否，若匹配则退出

3.项目核心代码

3.1 进程管理

processManage()

```
void processManage()
{
    int i;
    printf("================================================================================\n\n");
    printf("                                Process Manager                                 \n\n");
    printf("================================================================================\n");
    printf("%9s      |%10s      |%12s    |%10s  \n", "PID", "name", "priority", "running?");
```

```
    printf("------------------------------------------------------------------------
--\n");
    for (i = 0; i < NR_TASKS + NR_PROCS; ++i)
    {
        /* 遍历进程列表，输出每个进程的pid，名字，优先级和运行状态 */
        if (proc_table[i].p_flags != FREE_SLOT)
            printf("%8d        %10s      %10d             %s\n", proc_table[i].pid, pro
c_table[i].name, proc_table[i].priority, proc_table[i].p_flags == FREE_SLOT ? "NO" : "Y
ES");
    }
    printf("========================================================================
==\n\n");
}
```

3.2 文件管理

3.2.1 文件管理系统功能部分

3.2.1.1 获取目录下文件

get_file()

```
PUBLIC int get_file(char* filename, struct inode* dir_inode,
                    u32 mode)
{
    if (filename[0] == 0)   /* path: "/" */
        return dir_inode->i_num;
     * Search the dir for the file.
     */
    int dir_blk0_nr = dir_inode->i_start_sect;
    int nr_dir_blks = (dir_inode->i_size + SECTOR_SIZE - 1) / SECTOR_SIZE;
    int nr_dir_entries =
      dir_inode->i_size / DIR_ENTRY_SIZE; /**
                           * including unused slots
                           * (the file has been deleted
                           * but the slot is still there)
                           */
    int i, j, m = 0;
    struct dir_entry * pde;
    int dev = dir_inode->i_dev;
    for (i = 0; i < nr_dir_blks; i++)
    {
        RD_SECT(dev, dir_blk0_nr + i);
        pde = (struct dir_entry *)fsbuf;
        for (j = 0; j < SECTOR_SIZE / DIR_ENTRY_SIZE; j++, pde++)
        {
            if (strcmp(filename, pde->name) == 0)
```

```
            {
                int result = pde->inode_nr;
                struct inode * pin = get_inode(dev, pde->inode_nr);
                if (pin->i_mode == mode || mode == 0)
                {
                    put_inode(pin);
                    return result;
                }
                put_inode(pin);
                RD_SECT(dev, dir_blk0_nr + i);
            }

            if (++m > nr_dir_entries)
                break;
        }
        if (m > nr_dir_entries) /* all entries have been iterated */
            break;
    }
    /* file not found */
    return 0;
}
```

strip_path()

```
PUBLIC int strip_path(char * filename, const char * pathname,
              struct inode** ppinode)
{
    const char * s = pathname;
    char * t = filename;
    struct inode * dir_inode = get_inode(ROOT_DEV, ROOT_INODE);
    struct inode * next_inode;

    if (s == 0)
        return -1;
    if (*s == '/' || *s == '$' || *s == '#')
        s++;
    if (*s == '/')
        s++;
    while (*s) /* check each character */
    {
        if (*s == '/')
        {
            *t = 0;
            int dev = dir_inode->i_dev;
            int dir_inode_nr = get_file(filename, dir_inode, I_DIRECTORY);
            next_inode = get_inode(dev, dir_inode_nr);
            put_inode(dir_inode);
            dir_inode = next_inode;
            t = filename;
```

```
            s++;
        }
        *t++ = *s++;
        /* if filename is too long, just truncate it */
        if (t - filename >= MAX_FILENAME_LEN)
            break;
    }
    *t = 0;
    *ppinode = dir_inode;
    return 0;
}
```

3.2.1.2 文件的创建与查找

create_file()

```
PRIVATE struct inode * create_file(char * path, int flags)
{
    u32 mode = I_REGULAR;
    if (path[0] == '$') mode = I_DIRECTORY;
    char filename[MAX_PATH];
    struct inode * dir_inode;
    if (strip_path(filename, path, &dir_inode) != 0)
        return 0;
    int inode_nr = alloc_imap_bit(dir_inode->i_dev);
    int free_sect_nr = alloc_smap_bit(dir_inode->i_dev,
                      NR_DEFAULT_FILE_SECTS);
    struct inode *newino = new_inode(dir_inode->i_dev, inode_nr,
                     free_sect_nr, mode);
    new_dir_entry(dir_inode, newino->i_num, filename);
    return newino;
}
```

search_file()

```
PUBLIC int search_file(char * path)
{
    char * _path = path;
    if (path[0] == '$' || path[0] == '#') _path++;
    char filename[MAX_PATH];
    memset(filename, 0, MAX_FILENAME_LEN);
    struct inode * dir_inode;
    if (strip_path(filename, _path, &dir_inode) != 0)
        return 0;
    if (path[0] == '$')
    {
        return get_file(filename, dir_inode, I_DIRECTORY);
    }
    if (path[0] == '#')
```

```
    {
        return get_file(filename, dir_inode, I_REGULAR);
    }
    return get_file(filename, dir_inode, 0);
}
```

3.2.2 文件管理用户功能部分

3.2.2.1 ls指令实现

```
PUBLIC int do_ls()
{
    char pathname[MAX_PATH];
    /* get parameters from the message */
    int name_len = fs_msg.NAME_LEN; /* length of filename */
    int src = fs_msg.source;    /* caller proc nr. */
    assert(name_len < MAX_PATH);
    phys_copy((void*)va2la(TASK_FS, pathname),
          (void*)va2la(src, fs_msg.PATHNAME),
          name_len);
    pathname[name_len] = 0;
    char * filename;
    struct inode * dir_inode;
    strip_path(filename, pathname, &dir_inode);
    return get_list(dir_inode, I_DIRECTORY) + get_list(dir_inode, I_REGULAR);
}
```

3.2.2.2 mkdir指令实现

```
void mkdir(char * cur_dir, char * dir_name)
{
    //校验参数合法性
  if (dir_name[0] <= 0)
  {
    printf("mkdir:: error: command mkdir need one parameter.\n");
    return;
  }
  if (dir_name[0] == ' ')
  {
    printf("mkdir:: error: directory cannot start with space.\n");
    return;
  }
  char path[32] = "$";  //文件目录标识符
  int i = 0;
  for (; ; i++)
  {
    if (cur_dir[i] == 0) break;
    path[i + 1] = cur_dir[i];
```

```
  }
  i++;
  for (int j = 0; ; j++)
  {
    if (dir_name[j] == 0) break;
    path[i++] = dir_name[j];
  }
  path[i] = 0;
  int fd = open(path, O_CREAT);
  if (fd != -1)
  {
    close(fd);
  }
  else
  {
    printf("mkdir error: directory exists.\n");
  }
}
```

3.2.2.3 touch指令实现

```
void touch(char * cur_dir, char * filename)
{
    if (filename[0] <= 0)
    {
        printf("touch:: error: command touch need one parameter.\n");
        return;
    }
    if (filename[0] == ' ')
    {
        printf("touch:: error: file name cannot start with space.\n");
        return;
    }
    char path[32] = "#";   //普通目录标识符
    int i = 0;
    for (; ; i++)
    {
        if (cur_dir[i] == 0) break;
        path[i + 1] = cur_dir[i];
    }
    i++;
    for (int j = 0; ; j++)
    {
        if (filename[j] == 0) break;
        path[i++] = filename[j];
    }
    path[i] = 0;
    int fd = open(path, O_CREAT);
    if (fd != -1)
```

```
    {
        close(fd);
    }
    else
    {
        printf("touch error: file exists.\n");
    }
}
```

3.2.2.4 rm指令实现

```
void rm(char * cur_dir, char * filename, int flag)
{
    char path[32] = "#";

    if (flag == 1)
    {
        path[0] = '$';
    }
    int i = 0;
    for (; ; i++)
    {
        if (cur_dir[i] == 0) break;
        path[i + 1] = cur_dir[i];
    }
    i++;
    for (int j = 0; ; j++)
    {
        if (filename[j] == 0) break;
        path[i++] = filename[j];
    }
    path[i] = 0;
    int fd = unlink(path);
    if (fd == 0)
    {
        printf("%s deleted!\n", filename);
    }
    else
    {
        printf("Failed to delete %s!\n", filename);
    }
}
```

3.2.2.5 wt指令实现

```
void wt(char * cur_dir, char * filename)
{
  char path[32] = "#";
  int i = 0;
  for (; ; i++)
  {
    if (cur_dir[i] == 0) break;
    path[i + 1] = cur_dir[i];
  }
  i++;
  for (int j = 0; ; j++)
  {
    if (filename[j] == 0) break;
    path[i++] = filename[j];
  }
  path[i] = 0;
  int fd = open (path, O_RDWR);
  if (fd == -1)
  {
    printf("Failed to open %s!\n", filename);
    return;
  }
  char tty_name[] = "/dev_tty0";
  int fd_stdin  = open(tty_name, O_RDWR);
  if (fd_stdin == -1)
  {
    printf("Failed to write file!\n");
    return;
  }
  char writeBuf[4096];  // 写缓冲区
  int final = read(fd_stdin, writeBuf, 4096);
  writeBuf[final] = 0;
  write(fd, writeBuf, final + 1);
  printf("Totally write %d bytes in the file : %s\n",final,writeBuf);
  close(fd);
}
```

3.2.2.6 rd指令实现

```
void rd(char*cur_dir, char * filename){
  char path[32] = "#";
  int i = 0;
  for (; ; i++)
  {
    if (cur_dir[i] == 0) break;
    path[i + 1] = cur_dir[i];
```

```
  }
  i++;
  for (int j = 0; ; j++)
  {
    if (filename[j] == 0) break;
    path[i++] = filename[j];
  }
  path[i] = 0;
  int fd = open (path, O_RDWR);
  if(fd==-1){
    printf("Failed to open %s!\n", filename);
        return;
  }
  char buf[4096];
    int rdFlag = read(fd, buf, 4096);
    if (rdFlag == -1)  // 读取文件内容失败
    {
        printf("Failed to read file!\n");
        close(fd);
        return;
    }

    printf("Content of file %s :\n", filename);
    printf(" %s\n",buf);
    close(fd);
}
```

3.2.2.7 cd指令实现

```
void cd(char * cur_dir, char * dir_name)
{
    //处理特殊目录
    if (dir_name[0] <= 0)
    {
        printf("cd:: error: command cd need one parameter.\n");
        return;
    }
    if (dir_name[0] == ' ')
    {
        printf("cd:: error: directory cannot start with space.\n");
        return;
    }
    if (dir_name[0] == '.')
    {
        if (dir_name[1] == '.')
        {
            int i = 0;
            for (; ; i++)
            {
```

```
                if (cur_dir[i] == '/' && cur_dir[i + 1] == 0)
                {
                    break;
                }
            }
            if (i > 0)
            {
                cur_dir[i--] = 0;
                for (; ; i--)
                {
                    if (cur_dir[i] == '/') break;
                    cur_dir[i] = 0;
                }
            }
        }
        return;
    }
    char path[32] = "$";
    int i = 0;
    for (; ; i++)
    {
        if (cur_dir[i] == 0) break;
        path[i + 1] = cur_dir[i];
    }
    i++;
    for (int j = 0; ; j++)
    {
        if (dir_name[j] == 0) break;
        path[i++] = dir_name[j];
    }
    path[i] = 0;
    int fd = open(path, O_RDWR);
    if (fd == -1)
    {
        printf("No such directory.\n");
        return;
    }
    close(fd);
    strcat(cur_dir, dir_name);
    strcat(cur_dir, "/");
}
```

3.3 游戏及工具

3.3.1 游戏

3.3.1.1 sudoku（数独）

dfs(int i, int j, bool *success)

```
PUBLIC void dfs(int i, int j, bool *success)
{
    if (*success)
    {
        return;
    }

    if (i == 9 && j == 0)
    {
        /* 成功建立棋盘 */
        *success = true;
        copy_map();//复制棋盘
        return;
    }

    int next_i = j == 8 ? i + 1 : i;
    int next_j = j == 8 ? 0 : j + 1;

    /* 建立1-9的随机数组进行遍历 */
    for (int k = 9; k > 0; k--)
    {
        map_copy[i][j] = k;
        if (is_legal(i, j))//判断该处填入数字是否合法
        {
            dfs(next_i, next_j, success);//无限向下递归直到返回success
        }
        map_copy[i][j] = 0;//回溯
    }
}
```

is_legal(int i, int j)

```
/* 判断当前单元格填入数字后是否合法 */
PUBLIC bool is_legal(int i, int j)
{
    int start[9][2] = {{0, 0}, {0, 3}, {0, 6}, {3, 0}, {3, 3}, {3, 6}, {6, 0}, {6, 3},
{6, 6}};

    int st = get_start(i, j);
    int a = start[st][0], b = start[st][1];

    /* 九宫格内不能有重复数字 */
    for (int m = a; m < a + 3; m++)
```

```
    {
        for (int n = b; n < b + 3; n++)
        {
            if (m == i && n == j)
                break;
            if (map_copy[m][n] == map_copy[i][j])
                return false;
        }
    }

    for (int k = 0; k < i; k++)
    {
        /* 所属列不能有相同数字 */
        if (map_copy[k][j] == map_copy[i][j])
            return false;
    }

    for (int k = 0; k < j; k++)
    {
        /* 所属行不能有相同数字 */
        if (map_copy[i][k] == map_copy[i][j])
            return false;
    }
    return true;
}
```

sudoku_main(int *fd_stdin)

```
PUBLIC void sudoku_main(int *fd_stdin)
{
    /* init the game */
    get_map();

    char order[2];          // 指令
    char r[2], c[2], v[2]; // row,column,value

    /* press ENTER to start the game */
    sudoku_list();
    int start = read(*fd_stdin, order, 512);
    print_map();

    while (1)
    {
        printf("[E] to enter a number [D] to delete a number [Q] to quit the game [P]
to print the map\n");
        printf("Please enter the order: ");

        /* read */
        int p = read(*fd_stdin, order, 512);
```

```
        order[p] = 0;

        if (strcmp(order, "E") == 0)
        {
            /* 输入数字 */
            int row, col, val;
            printf("Please enter the row of the grid you want to complete: ");
            p = read(*fd_stdin, r, 512);
            r[p] = 0;
            row = r[0] - '0';

            printf("Please enter the column of the grid you want to complete: ");
            p = read(*fd_stdin, c, 512);
            c[p] = 0;
            col = c[0] - '0';

            if (status[row - 1][col - 1] != EMPTY)
            {
                printf("Sorry, grid[%d, %d] has been filled in.\n", row, col);
                print_map();
                continue;
            }

            printf("Please enter the number you want to fill in: ");
            p = read(*fd_stdin, v, 512);
            v[p] = 0;
            val = v[0] - '0';

            if (val == sudoku_map[row - 1][col - 1])
            {
                /* 输入正确 */
                status[row - 1][col - 1] = CORRECT;
            }
            else
            {
                /* 输入错误 */
                status[row - 1][col - 1] = WRONG;
            }
            player[row - 1][col - 1] = val;
        }
        else if (strcmp(order, "D") == 0)
        {
            /* 删除数字 */
            int row, col;
            printf("Please enter the row of the grid you want to delete: ");
            p = read(*fd_stdin, r, 512);
            r[p] = 0;
            row = r[0] - '0';
```

```
            printf("Please enter the column of the grid you want to delete: ");
            p = read(*fd_stdin, c, 512);
            c[p] = 0;
            col = c[0] - '0';

            if (status[row - 1][col - 1] == GIVEN)
            {
                /* 初始给定数字不能删除 */
                printf("Sorry, Grid[%d, %d] cannot be deleted.\n\n", row, col);
                continue;
            }
            else
            {
                status[row - 1][col - 1] = EMPTY;
                player[row - 1][col - 1] = 0;
            }
        }
        else if (strcmp(order, "Q") == 0)
        {
            printf("\n\n\n");
            break;
        }
        else if (strcmp(order, "P") != 0)
        {
            printf("False Command!\n\n");
            continue;
        }

        printf("\n\n\n\n\n\n\n\n\n\n");
        print_map();

        /* 判定是否获胜 */
        if (check_win())
        {
            printf("            Congratulation!! You successfully complete the
sudoku!!\n");
            break;
        }
    }
}
```

3.3.1.2 bwchess（黑白棋）

Exa(char board[][MAXSIZE], int arrput[][MAXSIZE], char level)

```
int Exa(char board[][MAXSIZE], int arrput[][MAXSIZE], char level)
{
    int rowd, cold, row, col, x, y = 0;
    int step = 0;
```

```
    char foe;
    if (level == 1)
        foe = -1; //对手
    else
        foe = 1;
    char player = -1 * foe;
    for (row = 0; row < MAXSIZE; row++)
    {
        for (col = 0; col < MAXSIZE; col++)
        {
            arrput[row][col] = 0;
        }
    }
    for (row = 0; row < MAXSIZE; row++)//循环判断棋盘中的哪些单元格可以下子
    {
        for (col = 0; col < MAXSIZE; col++)
        {
            if (board[row][col] != 0)//若棋盘上的对应位置不为空(表示已经有子)
                continue;
            for (rowd = -1; rowd <= 1; rowd++)//循环检查上下行
                for (cold = -1; cold <= 1; cold++)//循环检查左右列
                {
                    //判断是否越界
                    if (row + rowd < 0 || row + rowd >= MAXSIZE || col + cold < 0 ||
col + cold >= MAXSIZE || (rowd == 0 && cold == 0))
                    {
                        continue;
                    }
                    if (board[row + rowd][col + cold] == foe)
                    {
                        x = row + rowd;
                        y = col + cold;
                        while (1)//以对手的下子为起始点，沿着该方向查找自己方的棋子，以攻击对方棋
子
                        {
                            x += rowd;//对手下子的四周坐标
                            y += cold;
                            if (x < 0 || x >= MAXSIZE || y < 0 || y >= MAXSIZE)
                            {
                                break;
                            }
                            if (board[x][y] == 0)
                            {
                                break;
                            }
                            if (board[x][y] == player)
                            {
                                arrput[row][col] = 1;
                                step++;
```

```
                                break;
                            }
                        }
                    }
                }
        }
    }
    return step;//返回可下的位置数量（若返回值为0，表示没地方可下）
}
```

Hint(char board[][MAXSIZE], int arrput[][MAXSIZE], char level)

```
int Hint(char board[][MAXSIZE], int arrput[][MAXSIZE], char level) //最佳走法
{
    int row, col, i, j;
    char board1[MAXSIZE][MAXSIZE] = { 0 };
    int maxscore = 0;
    int score = 0;
    char foe;
    if (level == 1)
        foe = -1;
    else
        foe = 1;
    for (row = 0; row < MAXSIZE; row++)//考虑能下子的每种情况
        for (col = 0; col < MAXSIZE; col++)
        {
            if (!arrput[row][col])
                continue;
            for (i = 0; i < MAXSIZE; i++)
                for (j = 0; j < MAXSIZE; j++)
                {
                    board1[i][j] = board[i][j];
                }
            Print(board1, row, col, level);
            score = CalSore(board1, level);
            if (maxscore < score)
                maxscore = score;
        }
    return maxscore;
}
```

bwchess_main(int* fd_stdin)

```
int bwchess_main(int* fd_stdin)
{
    char board[MAXSIZE][MAXSIZE];
    int arrput[MAXSIZE][MAXSIZE] = { 0 };
```

```
    int row, col, x, y;
    int count = 0;
    int level = 0;
    int cross = 0;
    int score[2];
    char ok;

    int order = 0;
    int v, m = 1;
    //初始界面

    bwchess_list();
    printf("master@SHAKOS:~/bwchess $ ");

    char o[2], r[2], c[2]; // order,row,column
    int p = read(*fd_stdin, o, 512);
    o[p] = 0;
    v = o[0] - '0';

    if (v == 3)
        return 0;
    if (v == 1)
    {

        bwchess_intro();
        p = read(*fd_stdin, o, 512);
        v = 2;
    }
    if (v == 2)
    {

        clear();
        printf("=======================AI BW-CHESS=======================\n\n");

        printf("||     you will be the black one- ( X )              ||\n");
        printf("||     i will be the white one- ( O )                ||\n");


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

        printf(" enter 1 to be the fisrt, enter 0 to be the later : ");
        p = read(*fd_stdin, o, 512);
        o[p] = 0;
        order = o[0] - '0';

        if (order == 0)
            level = 1;
        if (level == 0)
        {
```

```
        level = 1;
    }
    else
    {
        level = 0;
    }
    count = 4;
    for (row = 0; row < MAXSIZE; row++) //棋盘初始
    {
        for (col = 0; col < MAXSIZE; col++)
        {
            board[row][col] = 0;
        }
    }
    board[MAXSIZE / 2 - 1][MAXSIZE / 2 - 1] = board[MAXSIZE / 2][MAXSIZE / 2] = -1;
    board[MAXSIZE / 2 - 1][MAXSIZE / 2] = board[MAXSIZE / 2][MAXSIZE / 2 - 1] = 1;
    Show(board);
    int i = 0;
    for (i = count; count < (MAXSIZE * MAXSIZE) && cross < 2;)
    {
        if (level == 1)
        {
            level = 0;
            if (Exa(board, arrput, 2))//判断是否有位置落子
            {
                while (1)
                {
                    printf("\nenter your action(r):");
                    p = read(*fd_stdin, r, 512);
                    r[p] = 0;

                    if (!strcmp(r, "Q"))
                    {
                        return;
                    }

                    x = r[0] - '0';

                    printf("\nenter your action(c):");
                    p = read(*fd_stdin, c, 512);
                    c[p] = 0;
                    y = c[0];

                    if (!strcmp(c, "Q"))
                    {
                        return;
                    }

                    x--;
```

```
                        if (y >= 'a')
                        {
                            y = y - 'a' + 1;
                        }
                        else
                        {
                            y = y - 'A' + 1;
                        }
                        y--;
                       //判断是否越界、是否有棋子
                        if (x >= 0 && y >= 0 && x < MAXSIZE && y < MAXSIZE && arrput[x]
[y])
                        {
                            Print(board, x, y, 2);
                            count++;
                            break;
                        }
                        else
                        {
                            printf("\nwrong order try again .\n");
                        }
                    }
                    Show(board); //更新棋盘
                }
                else if (++cross < 2)//无效下子的次数小于2
                {
                    printf("\n is my turn.\n");
                }
                else
                {
                    printf("\ngame over.\n");
                }
            }
            else
            {
                level = 1;
                if (Exa(board, arrput, 1))
                {
                    cross = 0;
                    Foeplay(board, arrput, 1);//下子
                    count++;
                    Show(board);
                }
                else
                {
                    if (++cross < 2)
                    {
                        printf("\ni cannot move,you turn\n");
                    }
```

```
                    else
                    {
                        printf("\ngame over.");
                    }
                }
            }
        }
        Show(board);
        score[0] = score[1] = 0;
        for (row = 0; row < MAXSIZE; row++) //计算分数
        {
            for (col = 0; col < MAXSIZE; col++)
            {
                score[0] = score[0] + (board[row][col] == -1);
                score[1] = score[1] + (board[row][col] == 1);
            }
        }
        printf("final score:\n");
        printf("white:%d\nblack:%d\n", score[1], score[0]);
        if (score[1] > score[0])
        {
            printf("\nha ha ha ,you lose \n\n");
        }
        else
        {
            printf("\noh,i will be back!\n\n");
        }
    }

    //scanf_s("%c", &ok);
}
```

Foeplay(char board[][MAXSIZE], int arrput[][MAXSIZE], char level)

```
void Foeplay(char board[][MAXSIZE], int arrput[][MAXSIZE], char level)//计算机思考落子位置
{
    int row, col, row1, col1, i, j;
    int score = 0, minscore = 100;
    char board1[MAXSIZE][MAXSIZE];//临时数组，保存棋盘的下子位置
    int arrput1[MAXSIZE][MAXSIZE];//临时数组，保存可下子的位置
    char foe;//对手下的棋子
    if (level == 1)
        foe = -1;
    else
        foe = 1;
    for (row = 0; row < MAXSIZE; row++)
    {
        for (col = 0; col < MAXSIZE; col++)
        {
```

```
            if (arrput[row][col] == 0)
            {
                continue;
            }
            for (i = 0; i < MAXSIZE; i++)
            {
                for (j = 0; j < MAXSIZE; j++)
                {
                    board1[i][j] = board[i][j];
                }
            }
            Print(board1, row, col, level);//试着在临时棋盘中的一个位置下子
            Exa(board1, arrput1, foe);//检查对手是否有地方可下子
            score = Hint(board1, arrput1, foe);//获得临时棋盘中对方下子的得分情况
            if ((row == 0 && col == 0) || (row == 0 && col == MAXSIZE - 1 || row ==
MAXSIZE - 1 && col == 0 || row == MAXSIZE - 1 && col == MAXSIZE - 1)) //保存对方得分最低的
下法
            {
                minscore = score;
                row1 = row;
                col1 = col;
            }
            else if (row == 0 || row == MAXSIZE - 1 || col == 0 || col == MAXSIZE - 1)
            {
                minscore = score;
                row1 = row;
                col1 = col;
            }
            else if (score < minscore)
            {
                minscore = score;
                row1 = row;
                col1 = col;
            }
        }
    }
    Print(board, row1, col1, level);//计算机按最优下法下子
}
```

3.3.1.3 tic-tac-toe（井字棋）

```
PUBLIC int ticTacToe_list();
PUBLIC int ticTacToe_main(int *fd_stdin)
{
    int player = 0;
    int winner = 0;
    int number = 0;
    int row = 0;
```

```
    int column = 0;


    char tic_tac_toe[3][3] = {
        {'1', '2', '3'},
        {'4', '5', '6'},
        {'7', '8', '9'}};

    //让双方玩家轮流输入自己想要标志的位置
    int i;
    char o[2];

    ticTacToe_list();
    read(*fd_stdin, o, 512);
    clear();

    for (i = 0; i < 9 && winner == 0; i++)
    {
        printf("\n");
        printf("                         ");
        printf(" %c | %c | %c \n", tic_tac_toe[0][0], tic_tac_toe[0][1], tic_tac_toe[0]
[2]);
        printf("                         ");
        printf("---+---+---\n");
        printf("                         ");
        printf(" %c | %c | %c \n", tic_tac_toe[1][0], tic_tac_toe[1][1], tic_tac_toe[1]
[2]);
        printf("                         ");
        printf("---+---+---\n");
        printf("                         ");
        printf(" %c | %c | %c \n", tic_tac_toe[2][0], tic_tac_toe[2][1], tic_tac_toe[2]
[2]);
        player = i % 2 + 1;
        do
        {
            printf("\nif you want to exit, please input 0.\n");
            printf("\nplayer %c", (player == 1) ? 'A' : 'B');
            printf(" input a number of the grid to put your %c piece:", (player == 1) ?
'X' : 'O');
            int p = read(*fd_stdin, o, 512);
            o[p] = 0;
            number = o[0] - '0';
            if(number==0) return 0;
            row = (number - 1) / 3;     //行的索引码
            column = (number - 1) % 3; //列的索引码
        } while (number < 0 || number > 9 || tic_tac_toe[row][column] > '9');
        tic_tac_toe[row][column] = (player == 1) ? 'X' : 'O';
        //检查此玩家是否获胜
        //检查对角线上该玩家是否获胜
```

```
        if ((tic_tac_toe[0][0] == tic_tac_toe[1][1] && tic_tac_toe[0][0] ==
tic_tac_toe[2][2]) ||
            (tic_tac_toe[0][2] == tic_tac_toe[1][1] && tic_tac_toe[0][2] ==
tic_tac_toe[2][0]))
        {
            winner = player;
        }
        //检查横或竖上该玩家是否获胜
        else
        {
            for (int line = 0; line < 3; line++)
            {
                if ((tic_tac_toe[line][0] == tic_tac_toe[line][1] && tic_tac_toe[line]
[0] == tic_tac_toe[line][2]) ||
                    (tic_tac_toe[0][line] == tic_tac_toe[1][line] && tic_tac_toe[0]
[line] == tic_tac_toe[2][line]))
                {
                    winner = player;
                }
            }
        }
        clear();
    }
    //公布最后得分面板
    printf("\n");
    printf("                        ");
    printf(" %c | %c | %c \n", tic_tac_toe[0][0], tic_tac_toe[0][1], tic_tac_toe[0]
[2]);
    printf("                        ");
    printf("---+---+---\n");
    printf("                        ");
    printf(" %c | %c | %c \n", tic_tac_toe[1][0], tic_tac_toe[1][1], tic_tac_toe[1]
[2]);
    printf("                        ");
    printf("---+---+---\n");
    printf("                        ");
    printf(" %c | %c | %c \n", tic_tac_toe[2][0], tic_tac_toe[2][1], tic_tac_toe[2]
[2]);
    //打印最后胜利者结果
    if (winner == 0)
    {
        printf("\nWhat a pity! Nobody win the game!\n");
    }
    else
    {
        printf("\nCongratulation! The winner is %c!\n", (winner == 1) ? 'A' : 'B');
    }

    printf("\n\nPress ENTER to continue...\n");
```

```
    read(*fd_stdin, o, 512);

    return 0;
}

PUBLIC int ticTacToe_list()
{
    clear();
    printf("
==========================================================================\n");
    printf("      #                                  Welcome to
#\n");
    printf("      #                              Tic-Tac-Toe --- an easy and fun game
#\n");
    printf("      #
#\n");
    printf("      # + + + + + +
#\n");
    printf("      # +   +    +       + + + + +  $     Welcome to play tic-tac-toe!
#\n");
    printf("      # +  +  +   +     +   +    +  $
#\n");
    printf("      # + +     +  +   +  +   +  +  $     Have fun with your friends!
#\n");
    printf("      # +         + + + +      + +  $
#\n");
    printf("      # 0            +           +  $
#\n");
    printf("      #     + + + + + + + + + +  0  $    Press ENTER to start the game
#\n");
    printf("      #      + + + + + + + + + +      $
#\n");
    printf("      #       + + + + + + + + +       $
#\n");
    printf("      #                               $
#\n");
    printf("      #  #### #  #   #   #  # ####  $
#\n");
    printf("      #  #    #  #  # #  # #  #  #  $
#\n");
    printf("      #  #### ####  # #  ##   #  #  $
#\n");
    printf("      #     # #  # #   # # #  #  #
#\n");
    printf("      #  #### #  # #   # #  # ####
#\n");
    printf("      #                      TONGJI UNIVERSITY
#\n");
```

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

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

3.3.1.4 carrycraft（推箱子）

PUBLIC int carrycraft_main(int* fd_stdin)

```
PUBLIC int carrycraft_main(int* fd_stdin)
{
    char o[2];

    craft_list();
    read(*fd_stdin, o, 512);

    while (1)
    {

        drawmain();
        int p = read(*fd_stdin, o, 512);
        o[p] = 0;
        char push = o[0];
        int count, caw;
        for (int i = 0; i < 9; i++)
        {
            for (int j = 0; j < 11; j++)
            {
                if (map[i][j] == 2 || map[i][j] == 6)
                {
                    count = i;
                    caw = j;
                }
            }
        }
        if (push == 'w')
        {

            if (map[count - 1][caw] == 0 || map[count - 1][caw] == 4)
            {
                map[count][caw] -= 2;
                map[count - 1][caw] += 2;
            }
            else if (map[count - 1][caw] == 3 || map[count - 1][caw] == 7)
            {
                if (map[count - 2][caw] == 0 || map[count - 2][caw] == 4)
                {
                    map[count][caw] -= 2;
```

```
                map[count - 1][caw] -= 1;
                map[count - 2][caw] += 3;
            }
        }
    }

    if (push == 's')
    {
        if (map[count + 1][caw] == 0 || map[count + 1][caw] == 4)
        {
            map[count][caw] -= 2;
            map[count + 1][caw] += 2;
        }

        else if (map[count + 2][caw] == 0 || map[count + 2][caw] == 4)
        {
            if (map[count + 1][caw] == 3 || map[count + 1][caw] == 7)
            {
                map[count][caw] -= 2;
                map[count + 1][caw] -= 1;
                map[count + 2][caw] += 3;
            }
        }
    }

    if (push == 'a')
    {
        if (map[count][caw - 1] == 0 || map[count][caw - 1] == 4)
        {
            map[count][caw] -= 2;
            map[count][caw - 1] += 2;
        }

        else if (map[count][caw - 2] == 0 || map[count][caw - 2] == 4)
        {
            if (map[count][caw - 1] == 3 || map[count][caw - 1] == 7)
            {
                map[count][caw] -= 2;
                map[count][caw - 1] -= 1;
                map[count][caw - 2] += 3;
            }
        }

    }

    if (push == 'd')
    {
        if (map[count][caw + 1] == 0 || map[count][caw + 1] == 4)
        {
```

```
                map[count][caw] -= 2;
                map[count][caw + 1] += 2;
            }

            else if (map[count][caw + 2] == 0 || map[count][caw + 2] == 4)
            {
                if (map[count][caw + 1] == 3 || map[count][caw + 1] == 7)
                {
                    map[count][caw] -= 2;
                    map[count][caw + 1] -= 1;
                    map[count][caw + 2] += 3;
                }
            }
            //break;
        }

        if(push == 'q')
        {
            return;
        }
    }
    return 0;
}
```

int drawmain()

```
int drawmain()
{
    clear();
    int i, j;
    win(); //判断输赢
    for (i = 0; i < 9; i++)
    {
        printf("                    ");
        for (j = 0; j < 11; j++)
        {
            switch (map[i][j])
            {
            case 0:
                printf("  "); //空白的位置
                break;
            case 1:
                printf("W "); //墙
                break;
            case 2:
                printf("P "); //代表人
                break;
            case 3:
```

```
                printf("C "); //代表箱子
                break;
            case 4:
                printf("T "); //代表目标地址
                break;
            case 6:
                printf("P "); //代表人和目标地址重合
                break;
            case 7:
                printf("G "); //代表箱子和目标地址重合
                break;
            }
        }
        printf("\n");
    }
    return 0;
}
```

3.3.1.4 mine（扫雷）

`void InitBoard()`

```
void InitBoard(char board[ROWS][COLS], int rows, int cols, char set)//为数组初始化，设计展示的界面
{
  int i = 0;
  int j = 0;
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < cols; j++)
    {
      board[i][j] = set;//set可以更改
    }
  }
}
```

`void FindMine()`

```
void FindMine(char mine[ROWS][COLS], char show[ROWS][COLS], int row, int col, int fd_stdin)
{
  int x = 0;
  int y = 0;
  int win = 0;
  int p;
  char xn[2];
  char yn[2];
```

```
    printf("\nCurrent score is %d.\n", win);
    printf("You'll win when score reaches 71!\n\n");
  while (win < (row * col - EASYCOUNT))
  {
    printf("please input x:>");
    p = read(fd_stdin, xn, 512);
    printf("please input y:>");
    x=xn[0]-'0';
    p = read(fd_stdin, yn, 512);
    y=yn[0]-'0';
    if (x >= 1 && x <= row && y >= 1 && y <= col)
    {
      if (mine[x][y] == '1')
      {
        clear();
        printf("Boom!!!  Game over!\n");
        DispalyBoard(mine, row, col);
        break;
      }
      else
      {
                ShowMineCount(mine, show, row, col, x, y, &win);
        clear();
        DispalyBoard(show, row, col);
                printf("\nCurrent score is %d.\n", win);
                printf("You'll win when score reaches 71!\n\n");
      }
      if (win == row * col - EASYCOUNT)
      {
                clear();
                DispalyBoard(mine, row, col);
        printf("\nYou win! Congratulation!!\n\n");
      }
    }
    else
    {
      printf("wrong index:>\n");
    }
  }
}
```

3.3.2 工具

3.3.2.1 日历

weekday(int year, int month, int day)

```
PUBLIC int weekday(int year, int month, int day)
{
    int count;
    count = (year - 1) + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400 +
total_day(year, month, day);
  //计算某一天是星期几的蔡勒公式
    count %= 7;
    return count;
}
```

display_month(int year, int month, int day)

```
PUBLIC void display_month(int year, int month, int day)
{

    int i = 0, j = 1;
    int week, max;
    week = weekday(year, month, 1);//假设输入的是1号，计算该月第一天是周几
    max = max_day(year, month);
    printf("\n                  %d - %s", year, month_dis[month - 1]);
    printf("\n");
    printf("\n     S     M     T     W     T     F     S\n\n");
    for (i = 0; i < week; i++)
        printf("      ");
    for (j = 1; j <= max; j++)
    {
        printf("%6d", j);
        if (i % 7 == 6)
            printf("\n\n");
        i++;
    }
    printf("\n\n");
}
```

calendar_main(int *fd_stdin)

```
PUBLIC int calendar_main(int *fd_stdin)
{
    int year, month, day;
    char y[5], m[3], d[3]; // year, month, day

    calendar_intro();
```

```
    printf("Please enter year:");
    int p = read(*fd_stdin, y, 512);

    y[p] = 0;
    year = 0;
    int i = 0;
    while (i < p)
    {
        year *= 10;
        year += (y[i++] - '0');
    }

    printf("Please enter month:");
    p = read(*fd_stdin, m, 512);
    m[p] = 0;
    month = 0;
    i = 0;
    while (i < p)
    {
        month *= 10;
        month += (m[i++] - '0');
    }

    printf("Please enter day:");
    p = read(*fd_stdin, d, 512);
    d[p] = 0;
    day = 0;
    i = 0;
    while (i < p)
    {
        day *= 10;
        day += (d[i++] - '0');
    }

    if (month < 1 || month > 12 || day < 1 || day > 31)
    {
        printf("error!");
        return -1;
    }

    display_week(year, month, day);
    display_month(year, month, day);

    printf("\nPress ENTER to continue...\n");
    read(*fd_stdin, d, 512);

    return 0;
}
```

3.3.2.2 计算器

int advanceCalculate(char* origin_exp)

```
int advanceCalculate(char* origin_exp) {
  char exp[30] = "\0";
  int pos = 0;
  for (int i = 0; i < strlen(origin_exp); ++i) {
    if (isOp(origin_exp[i])) {
      exp[pos] = ' ';
      ++pos;
      exp[pos] = origin_exp[i];
      ++pos;
      exp[pos] = ' ';
      ++pos;
    }
    else if (isSingleNum(origin_exp[i])) {
      exp[pos] = origin_exp[i];
      ++pos;
    }
  }
  // 初始化
  num_clear();
  op_stack_clear();
  current = 0;

  struct C result[100];
  int index = 0;

  // 在表达式尾部添加结束标识符
  addEndTag(exp);

  op_stack_push('#');
  struct C elem = transNext(exp);
  while (!isEmpty_op()) {
    char ch = elem.data;

    if (elem.tag == 1) {    //如果是操作数, 直接读入下一个内容
      result[index] = elem;
      index++;
      elem = transNext(exp);
    }
    else if (elem.tag == 0) { //如果是操作符,判断ch的优先级priorityOutStack和当前栈顶操作符的优先级priorityInStack
      char topch = op_stack_top();
      if (priorityInStack(topch) < priorityOutStack(ch)) {    //当前操作符优先级大,将ch压栈,读入下一个内容
        op_stack_push(ch);
        elem = transNext(exp);
```

```
      }
      else if (priorityInStack(topch) > priorityOutStack(ch)) { //当前优先级小,推展并输出到
结果中
        struct C buf;
        buf.data = op_stack_pop();
        buf.tag = 0;
        result[index] = buf;
        index++;
      }
      else {
        if (op_stack_top() == '(') {  //如果退出的是左括号则读入下一个内容
          elem = transNext(exp);
        }
        op_stack_pop();
      }
    }
  }

  return calculate(result, index);
}
```

int calculate(struct C rlt[], int size)

```
int calculate(struct C rlt[], int size)
{
  num_clear();
  for (int i = 0; i < size; ++i) {
    if (rlt[i].tag == 1) {//如果是数字就压入栈中
      num_stack_push(rlt[i].data);
    }
    else {// 如果是操作符，就从栈中弹出两个数字做运算
      int right = num_stack_pop();
      int left = num_stack_pop();
      num_stack_push(operate(left, rlt[i].data, right));// 再将计算结果压入栈中
    }
  }
  return num_stack_pop();
}
```

PUBLIC int calculator_main(int *fd_stdin)

```
PUBLIC int calculator_main(int *fd_stdin)
{
    int option, num1, num2, result;
    char a[20], b[20]; // num1,um2

    calculator_list();
```

```
    read(*fd_stdin, a, 512);
    clear();

    do
    {
        option = get_option(fd_stdin);
        if (option == 0)
            break;
        if(option == 5){
            char exp[30]="\0";
            printf("please input an math expression:");
            int p = read(*fd_stdin, exp, 512);
            advcCal(exp);
        }else{
             do
            {
                printf("\nplease input a number:");
                int p = read(*fd_stdin, a, 512);
                a[p] = 0;
                num1 = 0;
                int i = 0;
                while (i < p)
                {
                    num1 *= 10;
                    num1 += (a[i++] - '0');
                }

                printf("\nplease input another number:");
                p = read(*fd_stdin, b, 512);
                b[p] = 0;
                num2 = 0;
                i = 0;
                while (i < p)
                {
                    num2 *= 10;
                    num2 += (b[i++] - '0');
                }

                if (option == 4 && num2 == 0)
                {
                    printf("\nsorry!! divid can not be 0");
                }
                else
                {
                    switch (option)
                    {
                    case 1:
                        result = num1 + num2;
                        break;
```

```
                    case 2:
                        result = num1 - num2;
                        break;
                    case 3:
                        result = num1 * num2;
                        break;
                    case 4:
                        result = num1 / num2;
                    }
                    print_result(num1, num2, result, option);
                }
            } while (option == 4 && num2 == 0);
        }
    } while (option != 0);

    return 0;
}
```

休眠功能

`sleep()`

```
void sleep(int *fd_stdin)
{
    srand(rand());
    clear();
    char str[6] = { 0 };
    int i;

    for (i = 0; i < 5; i++)
    {
        str[i] = rand() % ('z' - 'a' + 1) + 'a'; //生成要求范围内的随机数。
    }

    sleep_list(str);
    char instr[6]={"\0"};
    while (1)
    {
        int p = read(*fd_stdin, instr, 5);
        //printf("%s\n",instr);
        if (p == 0)
        {
            //strcpy(instr,"\0");
            clear();
            sleep_list(str);
            continue;
        }
        else
```

```
        {
            if (strcmp(str, instr) == 0)
            {
                return;
            }
            else
            {
             //strcpy(instr,"\0");
             clear();
                sleep_list(str);
                continue;

            }
        }
    }

}
```

4.成员分工

成员	分工
蒙俊杰1952292	日历、数独、黑白棋人机对战的程序设计
唐烁1952293	多级目录文件系统、各种文件操作的实现，开机动画的实现
谢宇翔1951708	进程管理，休眠功能，扫雷的程序设计
刘洪博1952350	整数计算器、推箱子、井字棋的程序设计