

# 操作系统小学期项目文档——Noobux

---

## 操作系统小学期项目文档——Noobux

1. 项目描述
  - 1.1 项目成员及分工
  - 1.2 项目选题
  - 1.3 项目开发环境
2. 功能展示
  - 2.1 开机动画
  - 2.2 用户登录登出
  - 2.3 控制台
  - 2.4 文件系统
    - 2.4.1 文件创建与删除
      - 2.4.1.1 文件的创建
      - 2.4.1.2 文件的删除
      - 2.4.1.3 文件的恢复
      - 2.4.1.4 目录的创建
      - 2.4.1.5 目录的删除
      - 2.4.1.6 清空回收站
    - 2.4.2 文件系统的访问与读写
      - 2.4.2.1 目录的访问
      - 2.4.2.2 写文件
      - 2.4.2.3 读文件
      - 2.4.2.4 文件目录内容的展示
    - 2.4.3 文件系统的保护
      - 2.4.3.1 访问保护区
      - 2.4.3.2 修改密码
    - 2.4.4 文件检索
  - 2.5 进程模块
    - 2.5.1 进入进程模块
    - 2.5.2 创建一个子进程
    - 2.5.3 阻塞一个进程
    - 2.5.3 恢复一个进程
    - 2.5.3 结束一个进程
  - 2.6 多终端
  - 2.7 游戏
    - 2.7.1 扫雷
    - 2.7.2 推箱子
3. 关键源码说明
  - 3.1 用户登录
  - 3.2 控制台
  - 3.3 文件系统
    - 3.3.1 文件系统的创建与删除
      - 3.3.1.1 创建文件函数
      - 3.3.1.2 删除文件函数
      - 3.3.1.3 创建目录函数
      - 3.3.1.4 删除目录函数
      - 3.3.1.5 恢复文件函数
      - 3.3.1.6 清空文件夹函数
    - 3.3.2 文件系统的访问与读写
      - 3.3.2.1 合并路径函数
      - 3.3.2.2 访问目录
      - 3.3.2.3 展示目录内容
      - 3.3.2.4 写文件

- 3.3.2.5 读文件
  - 3.3.3 文件的保护
  - 3.3.4 文件检索
- 3.4 进程模块
  - 3.4.1 进入进程模块
  - 3.4.2 创建子进程
  - 3.4.3 阻塞进程
  - 3.4.4 恢复进程
  - 3.4.5 结束进程
- 3.5 多终端
- 3.6 游戏
  - 3.6.1 扫雷
  - 3.6.2 推箱子

# 1. 项目描述

## 1.1 项目成员及分工

学号	姓名	分工	占比	考勤
1850217	杨煜	控制台，简易多终端，进程模块	20	全勤
1852714	陈磊	开机动画，文件系统，回收站	20	全勤
1852522	刘祎康	开机动画，文件系统，文件保护	20	全勤
1850953	徐炳昌	命令行控制台，用户登录登出，项目文档	20	全勤
1850231	姚凯楠	两个游戏，文件搜索	20	全勤

## 1.2 项目选题

a.完成《Orange`s：一个操作系统的实现》项目要求

难度	要求	实现
B	修改或者重新实现参考源码的一个或多个模块或完成部分实验内容。对参考源码的一个或多个模块进行修改或者重新实现，如可以重新实现其文件系统，新增代码量至少达到相关模块代码的一半	改进文件系统（密码保护、回收站、多级文件系统、文件检索）
C	在参考源码上实现系统级应用或完成少数实验内容。系统级应用是指与操作系统内核交互较多，如磁盘工具，控制台等。通过调用较多的系统 API 来实现。	实现命令行控制台，实现用户登录登出，实现进程管理模块，实现一个多终端下的简易控制台
D	在参考的源码上实现一个用户级应用或完成实验不够。用户级应用是指通过调用较少的系统 API 实现一个用户友好的应用程序。	两个小游戏（推箱子和扫雷）

## 1.3 项目开发环境

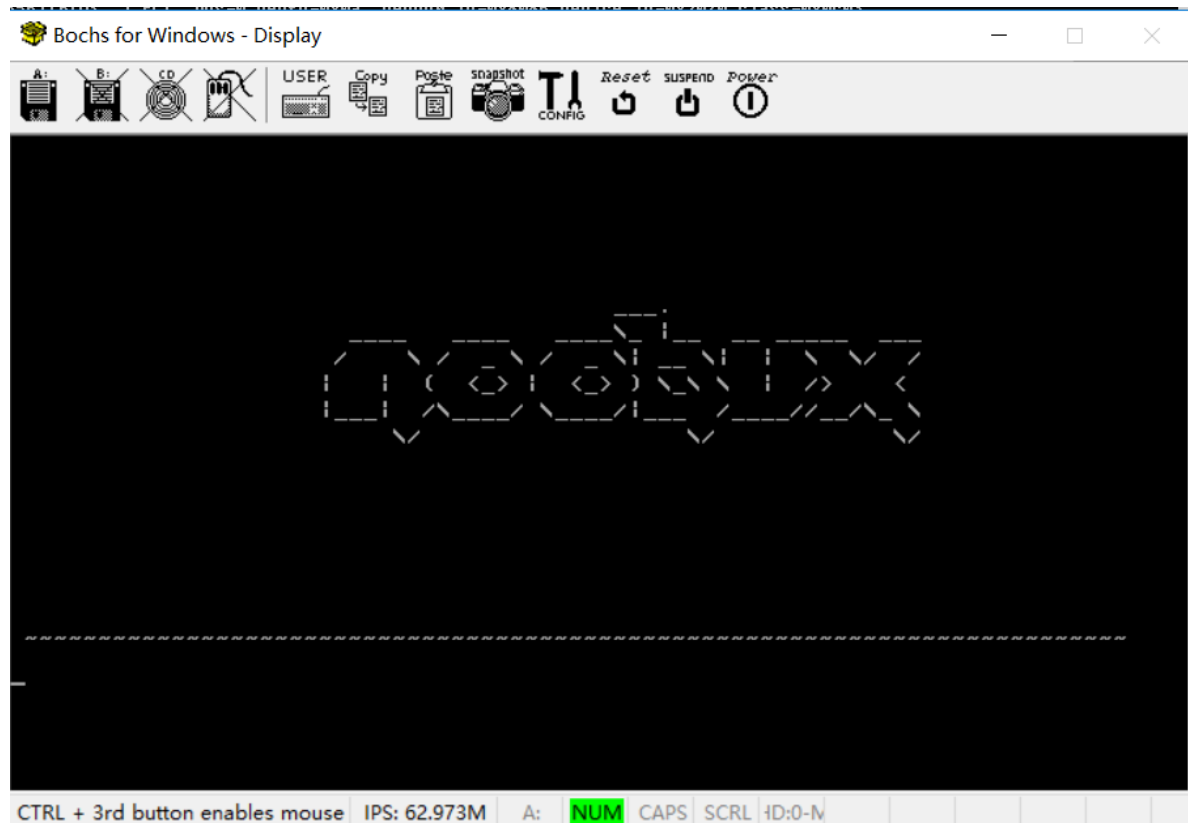
- ubuntu-20.04-desktop-amd64
- Bochs 2.6.11
- VMware-workstation-full-15.5.6-16341506

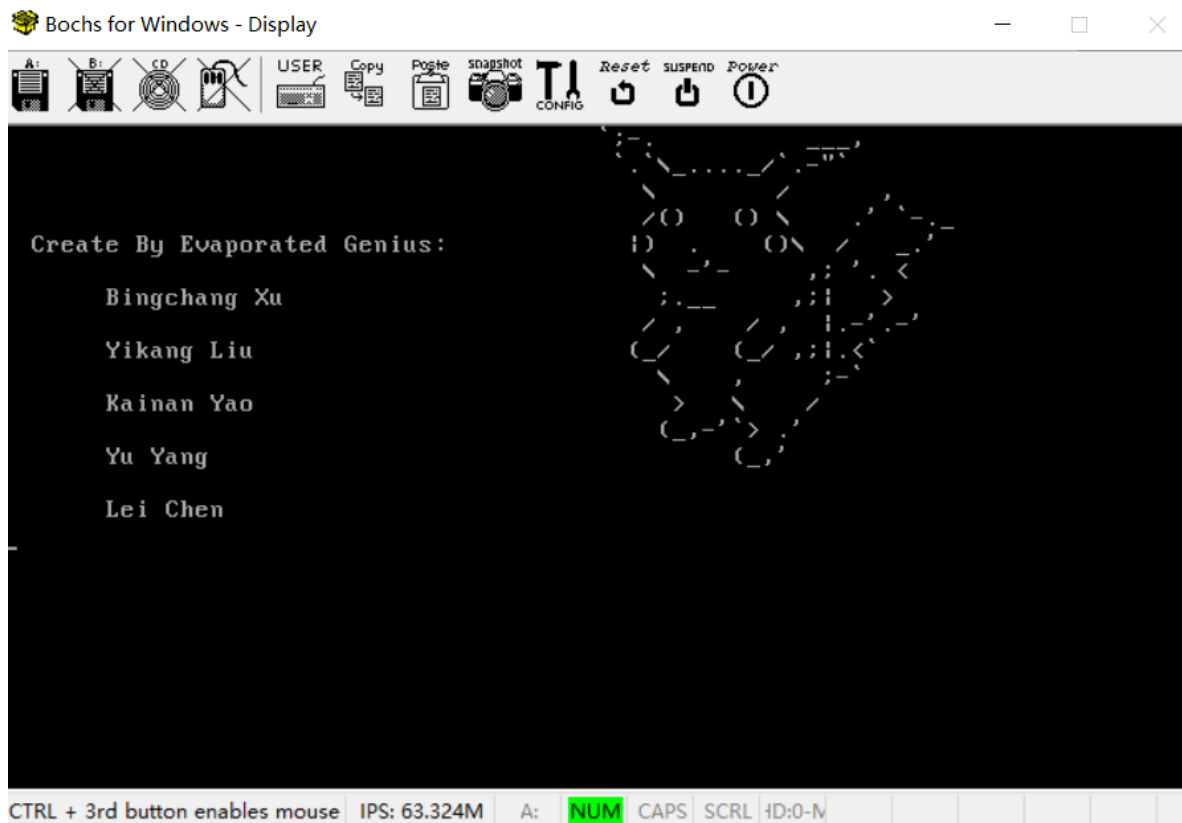
✈ 通过在VMware里运行Bochs运行本操作系统。

## 2. 功能展示

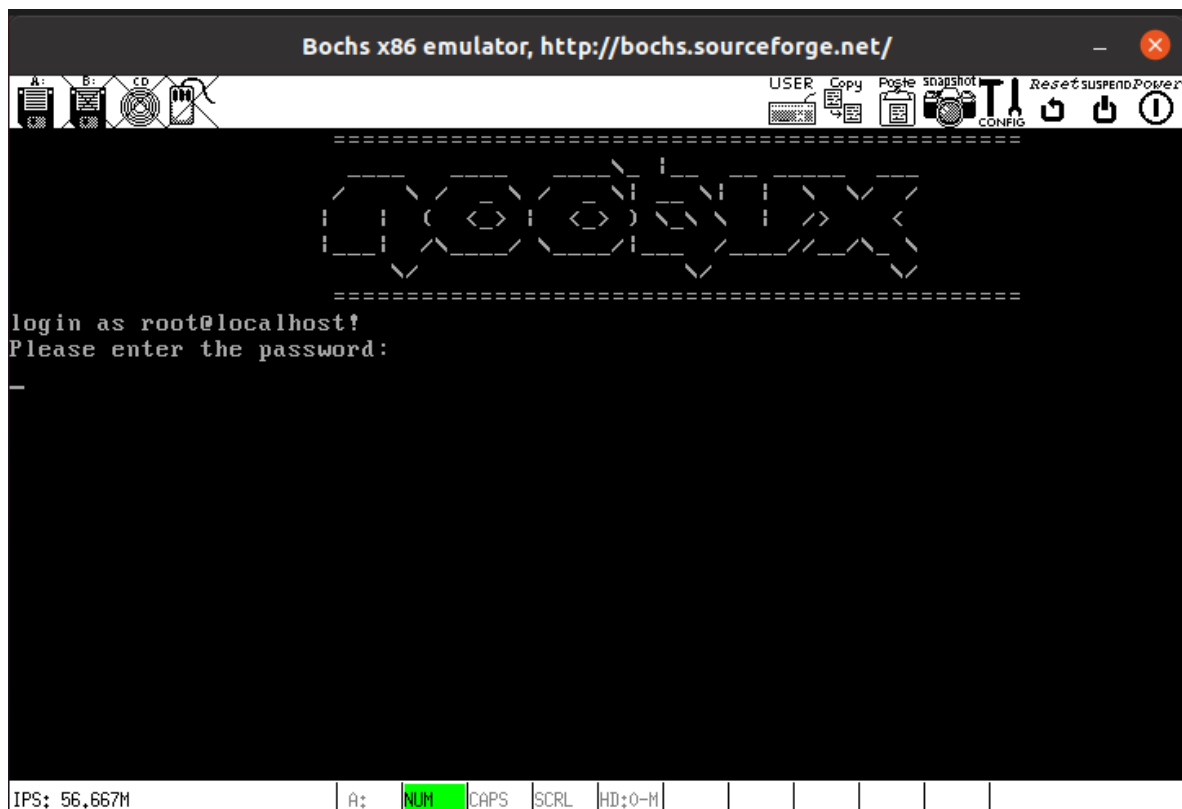
### 2.1 开机动画

noobux系统在开机时会先进入一段动画，这段动画为从右到左逐渐引入noobux的图标，之后从上到下逐渐消失。之后进入界面显示全体制作人员。在动画结束之后正式进入noobux操作系统





## 2.2 用户登录登出



用户通过输入密码来登录本操作系统，初始密码是123456，密码和受保护的文件的密码一致

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

=====
*                               Command List:
*                               Please Enter the command!
=====
*1. process                    | A Process Manager that display process information
*2. clear                      | Clear the Screen
*3. help                      | Show the Command List
*4. filemng                   | Start the File Management
*5. password                  | Check the password to access file
*6. ls                        | List all files in the current folder
*7. touch [filename]         | Create a new file in the current directory
*8. rmfile [filename]        | Delete a file in the current folder
*9. rsfile [filename]        | restore a file in the bin folder
*10. cat [filename]          | Output the content of a file in the current folder
*11. vi [filename]           | Apppend new content at the end of the file
*12. mkdir [foldername]      | Create a new floder in the current folder
*13. cd [foldername]         | Open a folder in the current folder
*14. runminesweep             | Run MineSweep game
*15. runmovebox               | Run MoveBox game
*16. clearbin                 | Clear the bin
*17. rmdir                    | Delete a directory in the current folder
*18. find [filename]         | Find the file with the input name
*19. logout                   | Exit current account
=====
[root@localhost: /]

IPS: 58.075M | A: NUM | CAPS | SCRL | HD: 0-M | | | | | |
```

用户可以在命令行中使用logout指令来退出登录，退出登录后再次输入密码可重新登录。

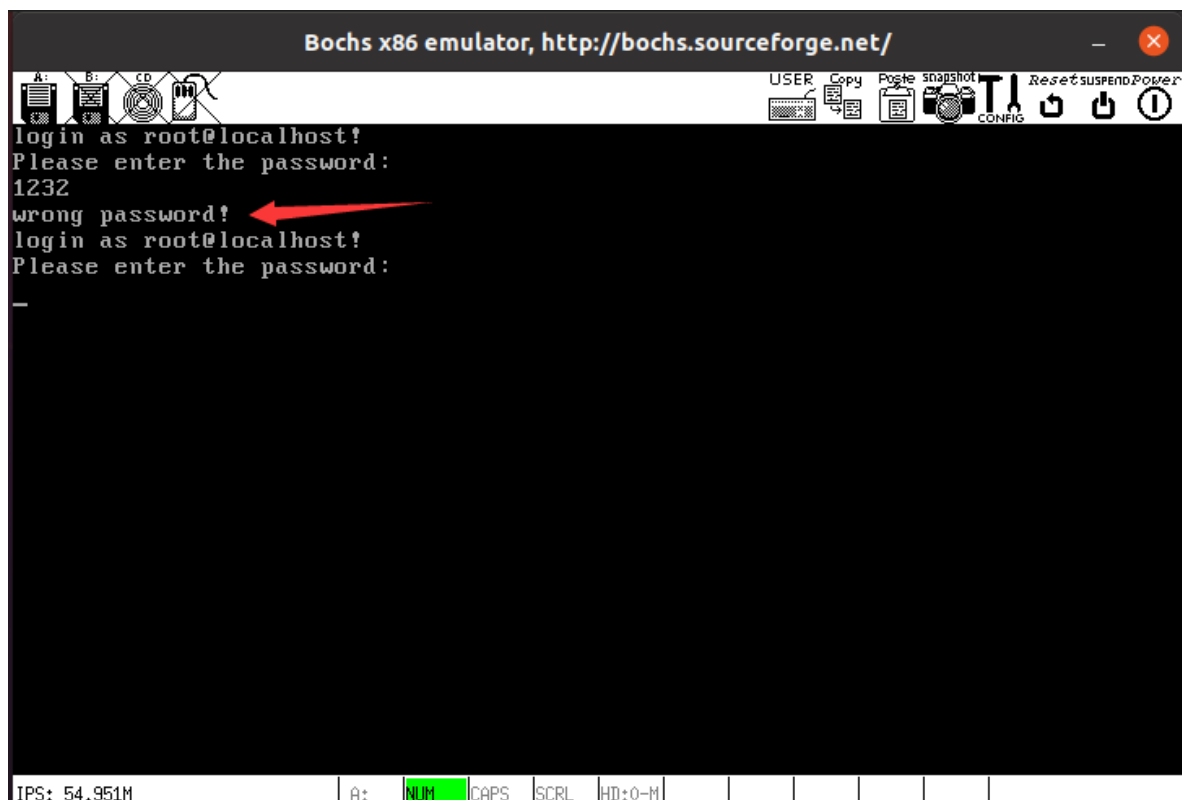
```
Bochs x86 emulator, http://bochs.sourceforge.net/

=====
login as root@localhost!
Please enter the password:

=====

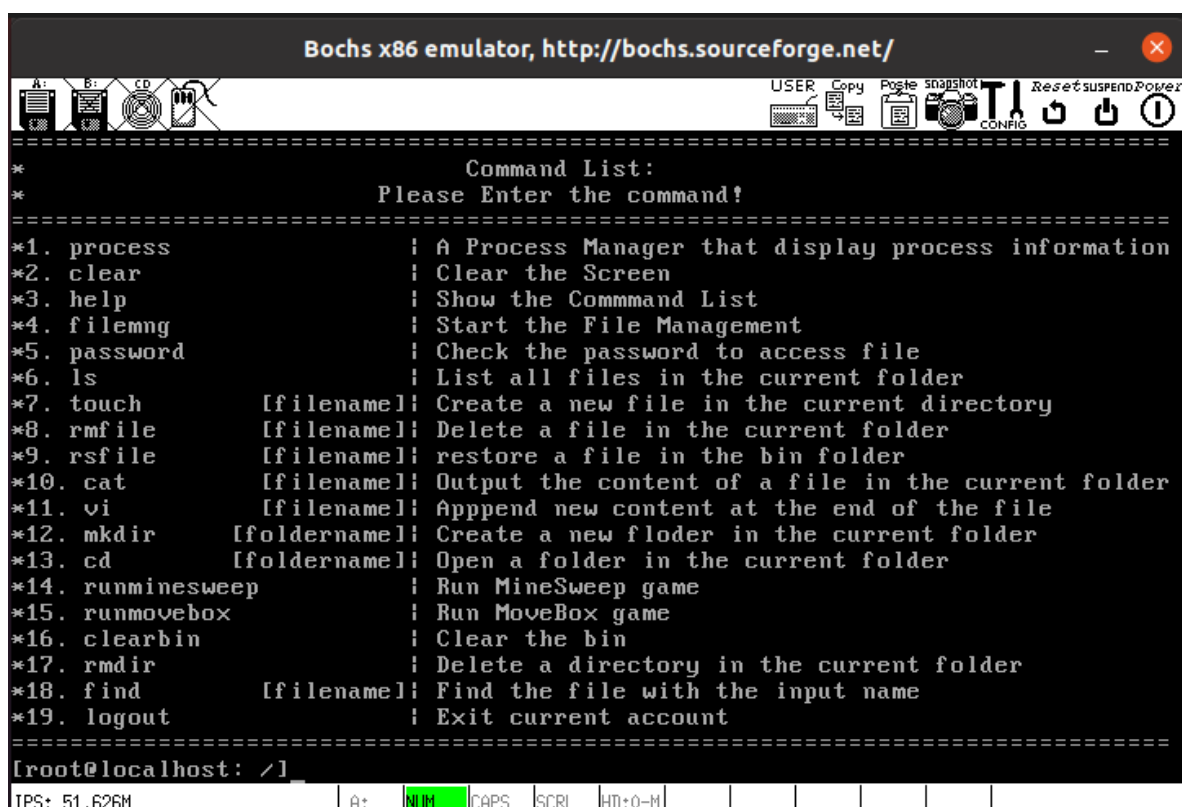
IPS: 46.673M | A: NUM | CAPS | SCRL | HD: 0-M | | | | | |
```

密码错误则会要求重新输入密码。



## 2.3 控制台

本操作系统的命令行详细如下：



在登录状态下输入对应的指令即可进行相应的操作。

## 2.4 文件系统

### 2.4.1 文件创建与删除

#### 2.4.1.1 文件的创建

在控制台中输入`touch+要创建的文件名`就可以在当前目录下创建文件，可以通过`vi`、`cat` 对文件做写、读操作

```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
=====
47 [dir] dir
54 file file
63 [dir] dir1
15 file password
60 file stone
=====
[root@localhost: /]touch txt
[root@localhost: /]ls

inode    type    filename
=====
1        file    .
2        file    dev_tty0
3        file    dev_tty1
4        file    dev_tty2
46       [dir]   bin
14       [dir]   protect
47       [dir]   dir
54       file    file
63       [dir]   dir1
15       file    password
64       file    txt
60       file    stone
=====
[root@localhost: /]
IPS: 80.573M  A: NUM CAPS SCRL HD:0-M
```

#### 2.4.1.2文件的删除

在控制台中输入`rmfile+要删除的文件名`就可以删除当前目录下的同名文件，如果在回收站bin中执行该操作，则文件会被彻底删除，而在其他目录下执行该操作，则会将该文件放入回收站中

在一般目录下删除文件：

```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
=====
3        file    dev_tty1
4        file    dev_tty2
46       [dir]   bin
14       [dir]   protect
47       [dir]   dir
54       file    file
15       file    password
60       file    stone
=====
[root@localhost: /]cat file
test
[root@localhost: /]rmfile file
successfully deleted!
File has moved to the bin
[root@localhost: /]cd bin
[root@localhost: /bin/]ls

inode    type    filename
=====
54       file    file
=====
[root@localhost: /bin/]cat file
test
[root@localhost: /bin/]la
IPS: 79.743M  A: NUM CAPS SCRL HD:0-M
```

在bin路径下删除文件：

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

USER Copy Paste snapshot CONFIG Reset suspend Power

```

4      file      dev_tty2
46     [dir]     bin
14     [dir]     protect
47     [dir]     dir
54     file      file
63     [dir]     dir1
15     file      password
60     file      stone
=====
[root@localhost: /]cd bin
[root@localhost: /bin/]ls

inode   type     filename
=====
64      file     txt
=====

[root@localhost: /bin/]rmfile txt
successfully deleted!
[root@localhost: /bin/]ls

inode   type     filename
=====
=====
[root@localhost: /bin/]

```

IPS: 74.951M | A: NUM | CAPS | SCRL | HD: 0-M | | | | |

### 2.4.1.3文件的恢复

如果在当前目录为bin时，在控制台输入`rsfile+要恢复的文件名`，就会将文件恢复至根目录下，如果根目录存在同名文件则无法恢复

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

USER Copy Paste snapshot CONFIG Reset suspend Power

```

54     file      file
=====
[root@localhost: /bin/]cat file
test
[root@localhost: /bin/]rsfile file
[root@localhost: /bin/]cd ..
[root@localhost: /]ls

inode   type     filename
=====
1       file     .
2       file     dev_tty0
3       file     dev_tty1
4       file     dev_tty2
46      [dir]    bin
14      [dir]    protect
47      [dir]    dir
54      file     file
15      file     password
60      file     stone
=====
[root@localhost: /]cat file
test
[root@localhost: /]_

```

IPS: 78.427M | A: NUM | CAPS | SCRL | HD: 0-M | | | | |

### 2.4.1.4目录的创建

在控制台中输入`mkdir+要创建的目录名`就可以在当前目录下创建相应的文件



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
=====
47 [dir] dir
54 file file
15 file password
60 file stone
=====
[root@localhost: /]mkdir dir
Failed to create a new directory with name dir
[root@localhost: /]mkdir dir1
[root@localhost: /]ls

inode    type    filename
=====
1        file    .
2        file    dev_tty0
3        file    dev_tty1
4        file    dev_tty2
46       [dir]    bin
14       [dir]    protect
47       [dir]    dir
54       file    file
63       [dir]    dir1
15       file    password
60       file    stone
=====
[root@localhost: /]
```

#### 2.4.1.5目录的删除

在控制台中输入`rmdir`+要删除的目录名 就可以删除当前目录下的同名目录

```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
=====
14 [dir] protect
47 [dir] dir
54 file file
63 [dir] dir1
15 file password
60 file stone
=====
[root@localhost: /]rmdir dir1
successfully deleted!
[root@localhost: /]ls

inode    type    filename
=====
1        file    .
2        file    dev_tty0
3        file    dev_tty1
4        file    dev_tty2
46       [dir]    bin
14       [dir]    protect
47       [dir]    dir
54       file    file
15       file    password
60       file    stone
=====
[root@localhost: /]la
```

#### 2.4.1.6清空回收站

在控制台输入`clearbin` 就可以清空回收站中所有文件

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

4      file      dev_tty2
46     [dir]     bin
14     [dir]     protect
47     [dir]     dir
54     file      file
63     [dir]     dir1
15     file      password
60     file      stone
=====
[root@localhost: ~]# cd bin
[root@localhost: ~]# bin/ls

inode    type      filename
=====
65       file      test
=====
[root@localhost: ~]# bin/clearbin
successfully clear the bin!
[root@localhost: ~]# bin/ls

inode    type      filename
=====
=====
[root@localhost: ~]# bin/

IPS: 79.870M  A: NUM CAPS SCRL HD:0-M
```

## 2.4.2 文件系统的访问与读写

### 2.4.2.1 目录的访问

在控制台中输入 `cd+要访问的目录名` 就可以进入对应的目录，当前所处目录会在localhost后显示

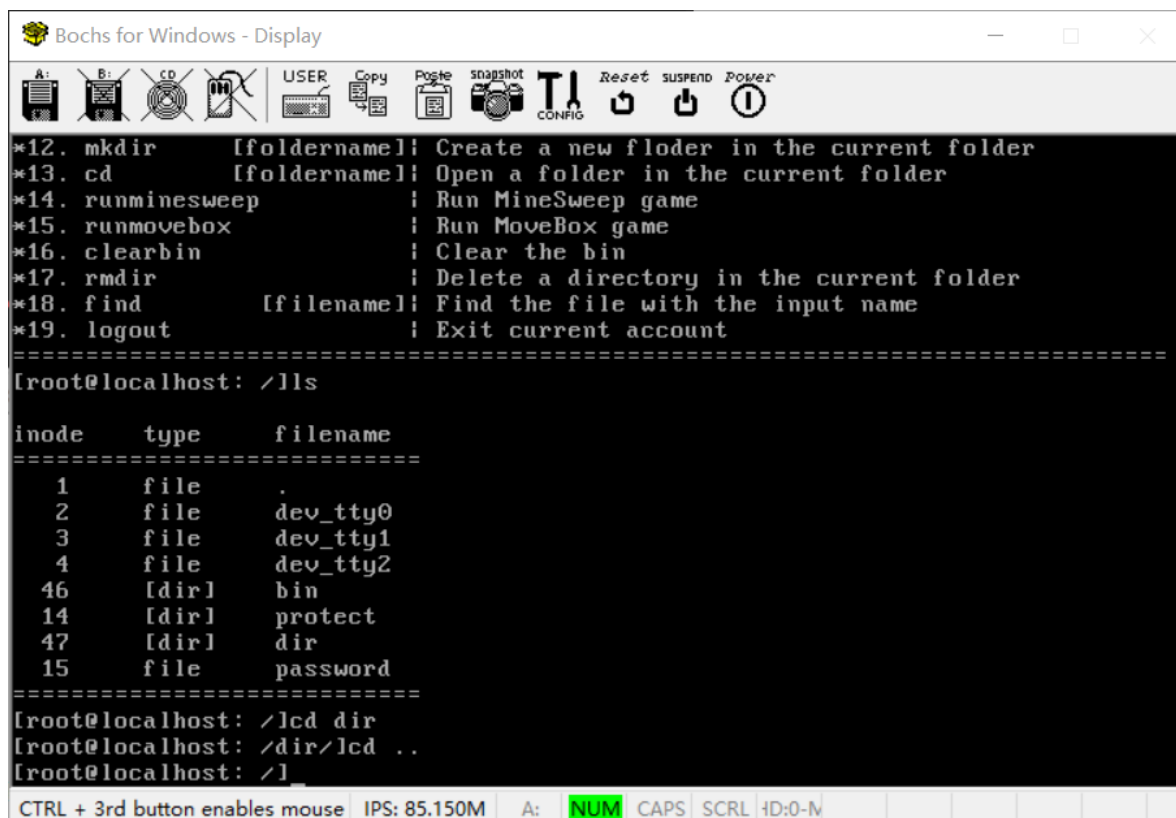
```
Bochs for Windows - Display

*11. vi      [filename]: Append new content at the end of the file
*12. mkdir   [foldername]: Create a new folder in the current folder
*13. cd      [foldername]: Open a folder in the current folder
*14. runminesweep : Run Minesweep game
*15. runmovebox : Run MoveBox game
*16. clearbin : Clear the bin
*17. rmdir   : Delete a directory in the current folder
*18. find    [filename]: Find the file with the input name
*19. logout  : Exit current account
=====
[root@localhost: ~]# ls

inode    type      filename
=====
1        file      .
2        file      dev_tty0
3        file      dev_tty1
4        file      dev_tty2
46       [dir]     bin
14       [dir]     protect
47       [dir]     dir
15       file      password
=====
[root@localhost: ~]# cd dir
[root@localhost: ~]# dir/

CTRL + 3rd button enables mouse  IPS: 79.192M  A: NUM CAPS SCRL HD:0-M
```

如果需要返回上一级目录，则只需要输入 `cd ..` 即可



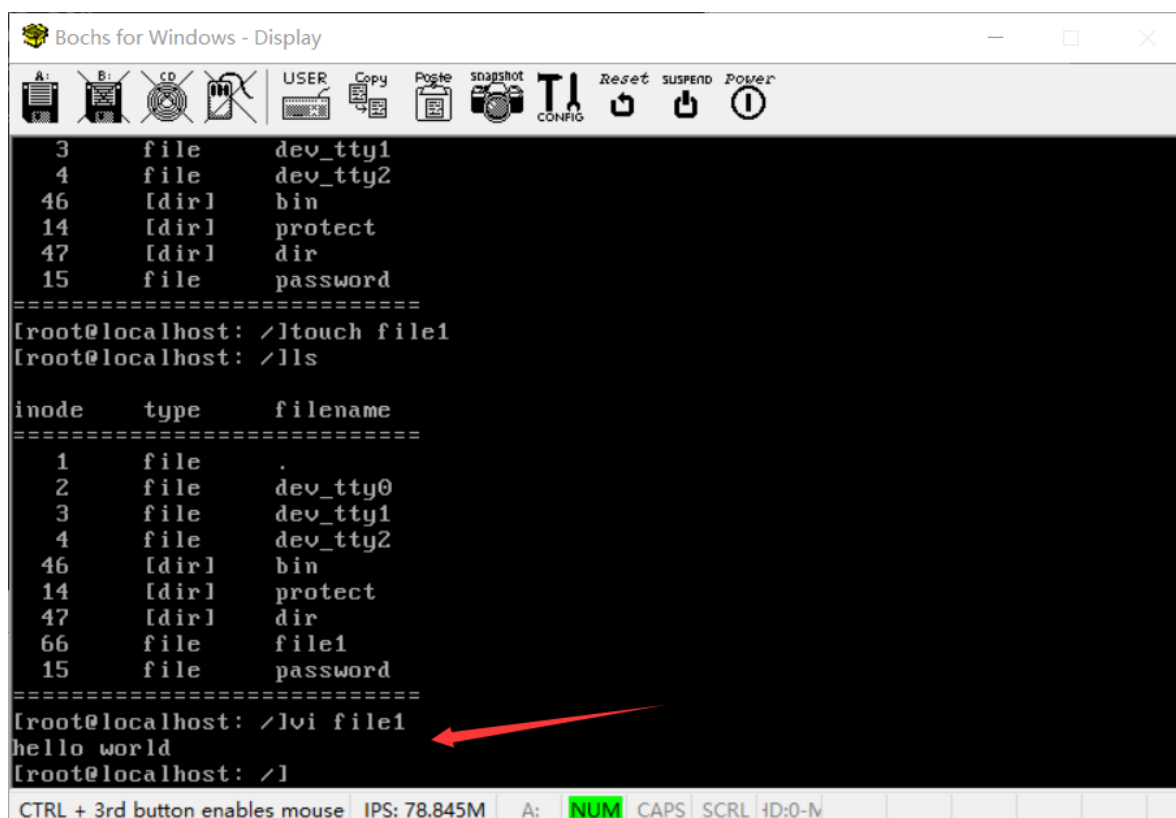
```
*12. mkdir      [foldername]! Create a new floder in the current folder
*13. cd         [foldername]! Open a folder in the current folder
*14. runminesweep      ! Run MineSweep game
*15. runmovebox        ! Run MoveBox game
*16. clearbin          ! Clear the bin
*17. rmdir             ! Delete a directory in the current folder
*18. find              [filename]! Find the file with the input name
*19. logout            ! Exit current account
=====
[root@localhost: ~]# ls

inode    type    filename
=====
   1      file      .
   2      file    dev_tty0
   3      file    dev_tty1
   4      file    dev_tty2
  46      [dir]    bin
  14      [dir]    protect
  47      [dir]    dir
  15      file    password
=====
[root@localhost: ~]# cd dir
[root@localhost: ~/dir]# cd ..
[root@localhost: ~]#
```

CTRL + 3rd button enables mouse IPS: 85.150M A: NUM CAPS SCRL ID:0-N

#### 2.4.2.2 写文件

输入 vi+ 文件名可以对文件进行写操作



```
   3      file    dev_tty1
   4      file    dev_tty2
  46      [dir]    bin
  14      [dir]    protect
  47      [dir]    dir
  15      file    password
=====
[root@localhost: ~]# touch file1
[root@localhost: ~]# ls

inode    type    filename
=====
   1      file      .
   2      file    dev_tty0
   3      file    dev_tty1
   4      file    dev_tty2
  46      [dir]    bin
  14      [dir]    protect
  47      [dir]    dir
  66      file    file1
  15      file    password
=====
[root@localhost: ~]# vi file1
hello world
[root@localhost: ~]#
```

CTRL + 3rd button enables mouse IPS: 78.845M A: NUM CAPS SCRL ID:0-N

#### 2.4.2.3 读文件

输入 cat+ 文件名可以对文件进行读操作

```
Bochs for Windows - Display
A: B: CD USER Copy Paste Snapshot CONFIG Reset Suspend Power
46 [dir] bin
14 [dir] protect
47 [dir] dir
15 file password
=====
[root@localhost: /]touch file1
[root@localhost: /]ls

inode    type    filename
=====
1        file    .
2        file    dev_tty0
3        file    dev_tty1
4        file    dev_tty2
46       [dir]   bin
14       [dir]   protect
47       [dir]   dir
66       file    file1
15       file    password
=====
[root@localhost: /]vi file1
hello world
[root@localhost: /]cat file1
hello world
[root@localhost: /]
```

CTRL + 3rd button enables mouse   IPS: 83.382M   A: NUM CAPS SCRL ID:0-N

#### 2.4.2.4 文件目录内容的展示

输入ls能够显示当前所在的目录中的文件以及目录

```
Bochs for Windows - Display
A: B: CD USER Copy Paste Snapshot CONFIG Reset Suspend Power
46 [dir] bin
14 [dir] protect
47 [dir] dir
66 file file1
15 file password
=====
[root@localhost: /]vi file1
hello world
[root@localhost: /]cat file1
hello world
[root@localhost: /]ls

inode    type    filename
=====
1        file    .
2        file    dev_tty0
3        file    dev_tty1
4        file    dev_tty2
46       [dir]   bin
14       [dir]   protect
47       [dir]   dir
66       file    file1
15       file    password
=====
[root@localhost: /]
```

CTRL + 3rd button enables mouse   IPS: 85.035M   A: NUM CAPS SCRL ID:0-N

### 2.4.3 文件系统的保护

#### 2.4.3.1 访问保护区

在根目录中有一个目录为protect，如果要对protect进行访问则需要输入密码，如果输入正确则可以进  
行访问，如果输入错误则不能进入protect目录

Bochs for Windows - Display

66 file file1  
15 file password  
=====

```
[root@localhost: /]vi file1  
hello world  
[root@localhost: /]cat file1  
hello world  
[root@localhost: /]ls
```

inode	type	filename
1	file	.
2	file	dev_tty0
3	file	dev_tty1
4	file	dev_tty2
46	[dir]	bin
14	[dir]	protect
47	[dir]	dir
66	file	file1
15	file	password

=====

```
[root@localhost: /]cd protect  
please enter your password  
123456  
[root@localhost: /protect/]
```

CTRL + 3rd button enables mouse IPS: 82.436M A: NUM CAPS SCRL ID:0-N

Bochs for Windows - Display

```
[root@localhost: /]cat file1  
hello world  
[root@localhost: /]ls
```

inode	type	filename
1	file	.
2	file	dev_tty0
3	file	dev_tty1
4	file	dev_tty2
46	[dir]	bin
14	[dir]	protect
47	[dir]	dir
66	file	file1
15	file	password

=====

```
[root@localhost: /]cd protect  
please enter your password  
123456  
[root@localhost: /protect/]cd ..  
[root@localhost: /]cd protect  
please enter your password  
123457  
wrong password  
[root@localhost: /]
```

CTRL + 3rd button enables mouse IPS: 84.411M A: NUM CAPS SCRL ID:0-N

#### 2.4.3.2 修改密码

在控制台中输入password可以对密码进行修改。在修改时需要先输入原密码，如果输入正确可以进行修改，如果输入错误则不能进行修改

```
Bochs for Windows - Display

=====
1    file    .
2    file    dev_tty0
3    file    dev_tty1
4    file    dev_tty2
46   [dir]   bin
14   [dir]   protect
47   [dir]   dir
66   file    file1
15   file    password
=====
[root@localhost: /]cd protect
please enter your password
123456
[root@localhost: /protect/]cd ..
[root@localhost: /]cd protect
please enter your password
123457
wrong password
[root@localhost: /]lpassword
please enter your password
123456
please enter new password
123
[root@localhost: /]_

CTRL + 3rd button enables mouse  IPS: 78.714M  A: NUM CAPS SCRL ID:0-N
```

```
Bochs for Windows - Display

=====
4    file    dev_tty2
46   [dir]   bin
14   [dir]   protect
47   [dir]   dir
66   file    file1
15   file    password
=====
[root@localhost: /]cd protect
please enter your password
123456
[root@localhost: /protect/]cd ..
[root@localhost: /]cd protect
please enter your password
123457
wrong password
[root@localhost: /]lpassword
please enter your password
123456
please enter new password
123
[root@localhost: /]lpassword
please enter your password
12
wrong password
[root@localhost: /]_

CTRL + 3rd button enables mouse  IPS: 80.417M  A: NUM CAPS SCRL ID:0-N
```

## 2.4.4 文件检索

利用cd指令跳转目录，利用ls指令展示当前目录，得到在根目录和dir文件夹下有文件号分别为60和63的名为stone两个文件

```
Bochs x86 emulator, http://bochs.sourceforge.net/
[root@localhost: /]ls
inode    type    filename
=====
1        file    .
2        file    dev_tty0
3        file    dev_tty1
4        file    dev_tty2
46       [dir]   bin
14       [dir]   protect
47       [dir]   dir
54       file    file
15       file    password
59       [dir]   test1
60       file    stone
=====
[root@localhost: /]cd dir
[root@localhost: /dir/]ls
inode    type    filename
=====
63       file    stone
=====
[root@localhost: /dir/]
IPS: 94.277M  A: NUM ICAPS SCRL HD:0-M
```

运行find stone，得到两个文件的文件号以及相应的位置

```
Bochs x86 emulator, http://bochs.sourceforge.net/
14       [dir]   protect
47       [dir]   dir
54       file    file
15       file    password
59       [dir]   test1
60       file    stone
=====
[root@localhost: /]cd dir
[root@localhost: /dir/]ls
inode    type    filename
=====
63       file    stone
=====
[root@localhost: /dir/]find stone
inode    location
=====
60       /stone
63       /dir/stone
=====
[root@localhost: /dir/]
IPS: 94.217M  A: NUM ICAPS SCRL HD:0-M
```

## 2.5 进程模块

### 2.5.1 进入进程模块

在控制台中输入命令process，即可进入进程模块之中。进入该模块之后会显示当前正在运行的进程的各个信息，以及在这个模块之中可以进行的四个操作的命令。

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

*19. logout          ! Exit current account
=====
[root@localhost: ~]# process
=====
      PID      |  name      |  spriority  |  running?
=====
      0         |  TTY       |      15     |      YES
      1         |  SYS       |      15     |      YES
      2         |  HD        |      15     |      YES
      3         |  FS        |      15     |      YES
      4         |  MM        |      15     |      YES
      5         |  TestA     |      5      |      YES
      6         |  TestB     |      5      |      YES
      7         |  TestC     |      5      |      YES
=====

*          Process Command List:
*          Please Enter the command!
=====
*1. kill      [pid]! Kill a process by its pid
*2. block     [pid]! Block a process by its pid
*3. unblock   [pid]! Unblock a process by its pid
*4. fork      ! Fork a process
=====
[root@localhost: ~]#
```

## 2.5.2 创建一个子进程

在控制台中输入命令fork，会以当前运行的进程作为父进程创建一个子进程，同时为这个子进程分配pid，并在进程模块中显示出来。

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

      6         |  TestB     |      5      |      YES
      7         |  TestC     |      5      |      YES
=====

*          Process Command List:
*          Please Enter the command!
=====
*1. kill      [pid]! Kill a process by its pid
*2. block     [pid]! Block a process by its pid
*3. unblock   [pid]! Unblock a process by its pid
*4. fork      ! Fork a process
=====
[root@localhost: ~]# fork
{MM} pid: 5
{MM} name: TestA
{MM} FFFF,0,0,0,F9,CF)
{MM} 100000
{MM} 80
{MM} 1000
{MM} 0
{MM} 400000
{MM} FFFF,0,0,0,F3,CF)
{MM} base: 0, limit: 1048575, size: 0)
{MM} childpid:8 0xA00000 <- 0x0 (0x100000 bytes)
[root@localhost: ~]# la
```

重新在控制台输入命令process，即可看到新建立的子进程。



```
Bochs x86 emulator, http://bochs.sourceforge.net/
[MM] childpid:8 0xA00000 <- 0x0 (0x100000 bytes)
[root@localhost: /]root@localhost: /]process
=====
PID      | name      | priority | running?
=====
0        | TTY       | 15       | YES
1        | SYS       | 15       | YES
2        | HD        | 15       | YES
3        | FS        | 15       | YES
4        | MM        | 15       | YES
5        | TestA     | 5        | YES
6        | TestB     | 5        | YES
7        | TestC     | 5        | YES
8        | TestA_8   | 5        | YES
=====

*          Process Command List:
*          Please Enter the command!
=====
*1. kill    [pid]: Kill a process by its pid
*2. block   [pid]: Block a process by its pid
*3. unblock [pid]: Unblock a process by its pid
*4. fork    | Fork a process
=====
[root@localhost: /]
IPS: 101.814M | A: NUM | CAPS | SCRL | HD:0-M |
```

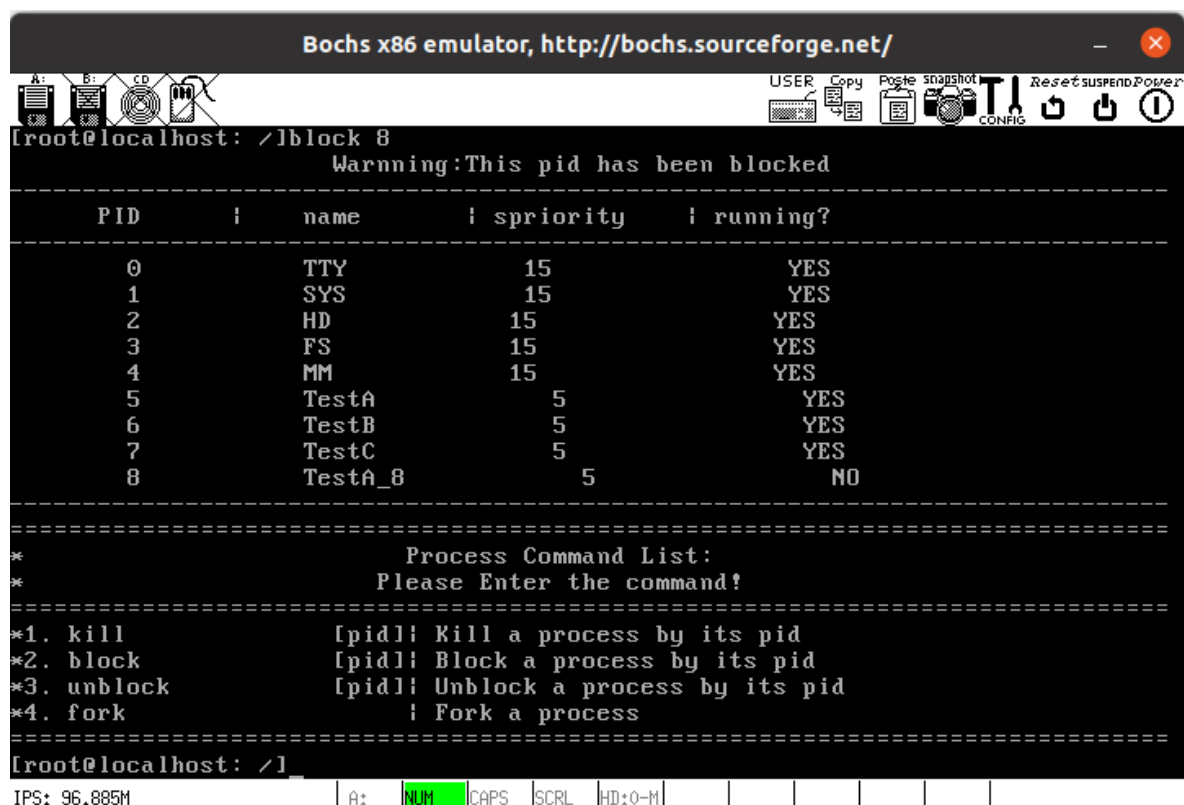
### 2.5.3 阻塞一个进程

在控制台中输入命令block+pid，即可将所要阻塞的进程，加以阻塞。如果输入的pid不合法，则给出相应的提示。

```
Bochs x86 emulator, http://bochs.sourceforge.net/
[root@localhost: /]block 8
Process TestA_8 has been blocked
=====
PID      | name      | priority | running?
=====
0        | TTY       | 15       | YES
1        | SYS       | 15       | YES
2        | HD        | 15       | YES
3        | FS        | 15       | YES
4        | MM        | 15       | YES
5        | TestA     | 5        | YES
6        | TestB     | 5        | YES
7        | TestC     | 5        | YES
8        | TestA_8   | 5        | NO
=====

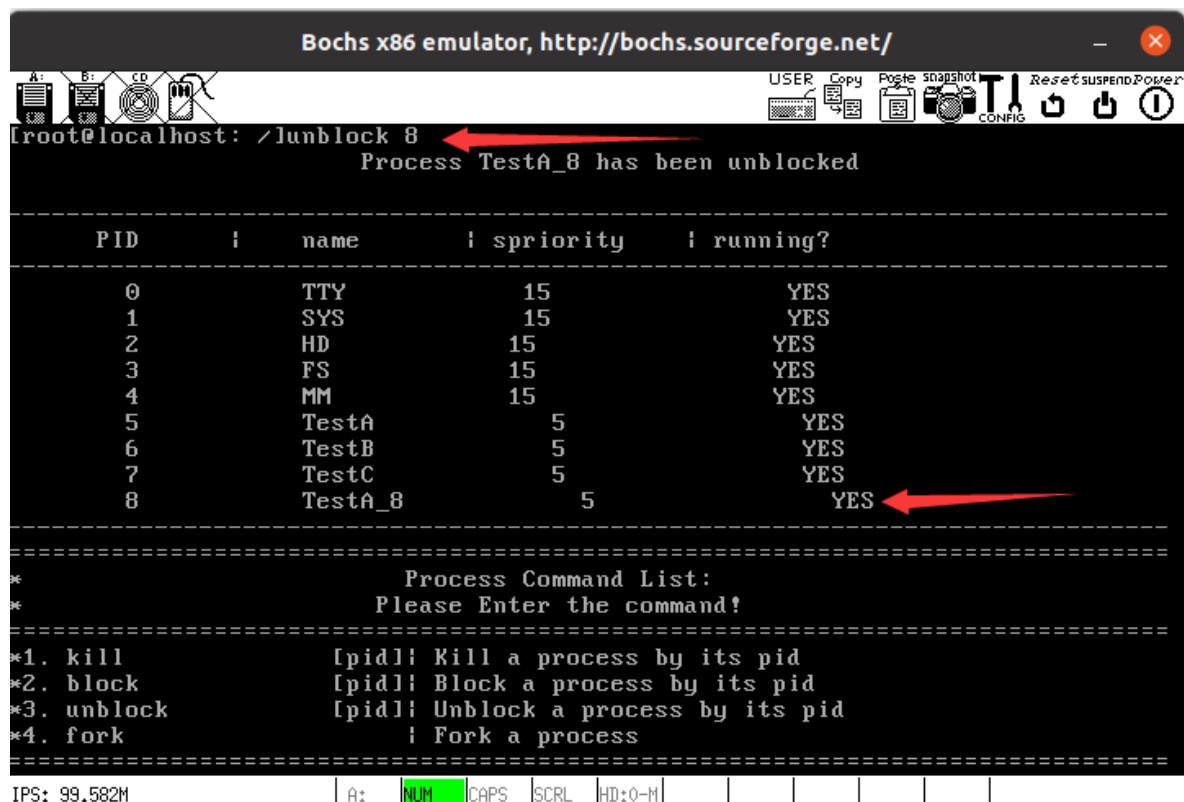
*          Process Command List:
*          Please Enter the command!
=====
*1. kill    [pid]: Kill a process by its pid
*2. block   [pid]: Block a process by its pid
*3. unblock [pid]: Unblock a process by its pid
*4. fork    | Fork a process
=====
IPS: 98.593M | A: NUM | CAPS | SCRL | HD:0-M |
```

如果再次阻塞已经被阻塞的进程则会进行提示。



### 2.5.3 恢复一个进程

在控制台中输入命令`unblock+pid`，即可将所要恢复的进程，加以恢复。如果输入的pid不合法，则给出相应的提示。



如果再次恢复已经被恢复的进程则会进行提示。

```
Bochs x86 emulator, http://bochs.sourceforge.net/
A: B: C: D: E: F: G: H: I: J: K: L: M: N: O: P: Q: R: S: T: U: V: W: X: Y: Z:
USER Copy Paste Snapshot CONFIG Reset Suspend Power

[root@localhost: ~]# ./unblock 8
Warning: This pid is still running

=====
PID      |  | name      |  | priority  |  | running?
=====
0        |  | TTY       |  | 15        |  | YES
1        |  | SYS       |  | 15        |  | YES
2        |  | HD        |  | 15        |  | YES
3        |  | FS        |  | 15        |  | YES
4        |  | MM        |  | 15        |  | YES
5        |  | TestA     |  | 5         |  | YES
6        |  | TestB     |  | 5         |  | YES
7        |  | TestC     |  | 5         |  | YES
8        |  | TestA_8   |  | 5         |  | YES
=====

=====
*          Process Command List:
*          Please Enter the command!
=====
*1. kill      [pid]: Kill a process by its pid
*2. block     [pid]: Block a process by its pid
*3. unblock   [pid]: Unblock a process by its pid
*4. fork      | Fork a process
=====

[root@localhost: ~]#
```

IPS: 97.469M | A: NUM | CAPS | SCRL | HD: 0-M | | | | | |

2.5.3 结束一个进程

在控制台中输入命令kill+pid，即可将所要结束的进程，加以结束。如果输入的pid不合法，则给出相应的提示。

```
Bochs x86 emulator, http://bochs.sourceforge.net/
A: B: C: D: E: F: G: H: I: J: K: L: M: N: O: P: Q: R: S: T: U: V: W: X: Y: Z:
USER Copy Paste Snapshot CONFIG Reset Suspend Power

[root@localhost: ~]# ./kill 8
Process TestA_8 has been killed

=====
PID      |  | name      |  | priority  |  | running?
=====
0        |  | TTY       |  | 15        |  | YES
1        |  | SYS       |  | 15        |  | YES
2        |  | HD        |  | 15        |  | YES
3        |  | FS        |  | 15        |  | YES
4        |  | MM        |  | 15        |  | YES
5        |  | TestA     |  | 5         |  | YES
6        |  | TestB     |  | 5         |  | YES
7        |  | TestC     |  | 5         |  | YES
=====

=====
*          Process Command List:
*          Please Enter the command!
=====
*1. kill      [pid]: Kill a process by its pid
*2. block     [pid]: Block a process by its pid
*3. unblock   [pid]: Unblock a process by its pid
*4. fork      | Fork a process
=====

[root@localhost: ~]#
```

IPS: 99.351M | A: NUM | CAPS | SCRL | HD: 0-M | | | | | |

如果再次结束已经被结束的进程则会进行提示。

```
Bochs x86 emulator, http://bochs.sourceforge.net/
=====
[root@localhost: ~]# kill 8
Warning: This pid has been killed
=====
      PID      | name      | priority | running?
-----|-----|-----|-----
      0      | TTY       | 15       | YES
      1      | SYS       | 15       | YES
      2      | HD        | 15       | YES
      3      | FS        | 15       | YES
      4      | MM        | 15       | YES
      5      | TestA     | 5        | YES
      6      | TestB     | 5        | YES
      7      | TestC     | 5        | YES
=====
*               Process Command List:
*               Please Enter the command!
=====
*1. kill        [pid]! Kill a process by its pid
*2. block       [pid]! Block a process by its pid
*3. unblock     [pid]! Unblock a process by its pid
*4. fork        ! Fork a process
=====
[root@localhost: ~]#
IPS: 96.606M  A: NUM CAPS SCRL HD:0-M
```

## 2.6 多终端

调用系统指令，完成了多终端的控制。键盘上键入Alt+F2，则切换到进程TestB，在这个进程下可以随意的进行键盘输入。

```
Bochs x86 emulator, http://bochs.sourceforge.net/
=====
[TTY #1]
$
=====
IPS: 99.613M  A: NUM CAPS SCRL HD:0-M
```

## 2.7 游戏

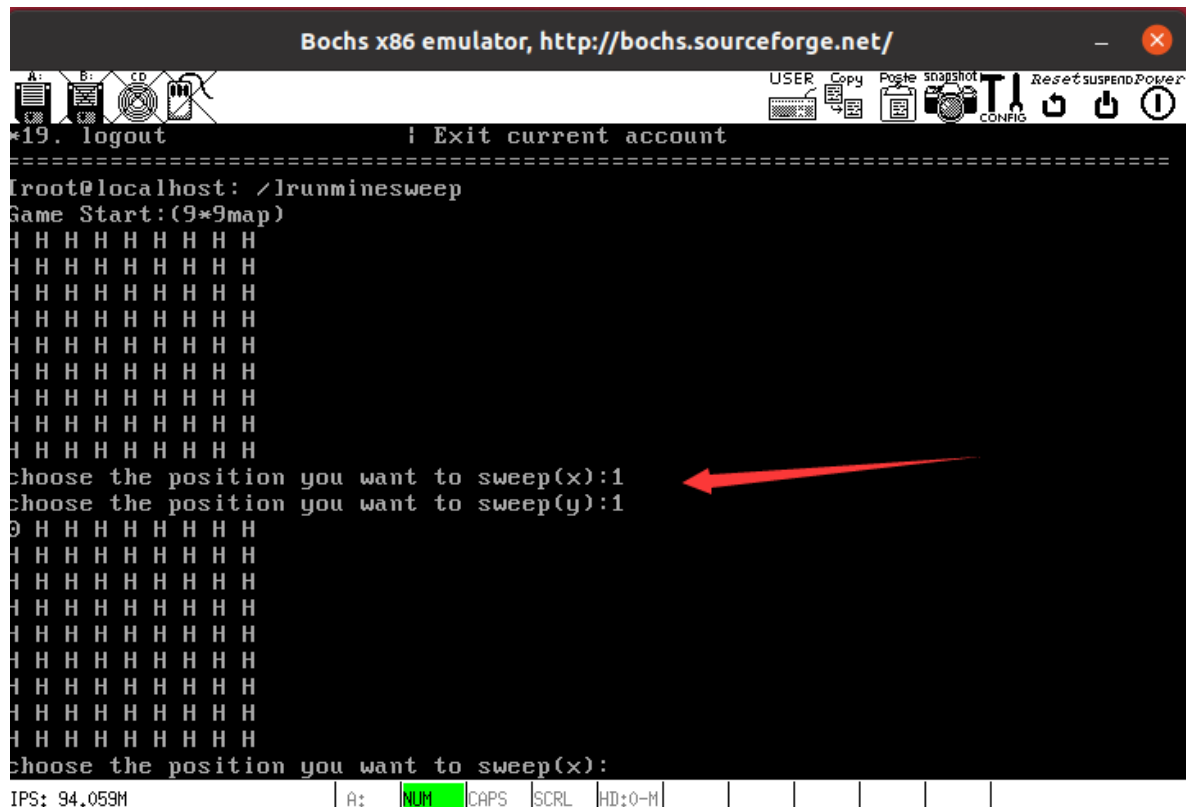
### 2.7.1 扫雷

规则与经典扫雷相同

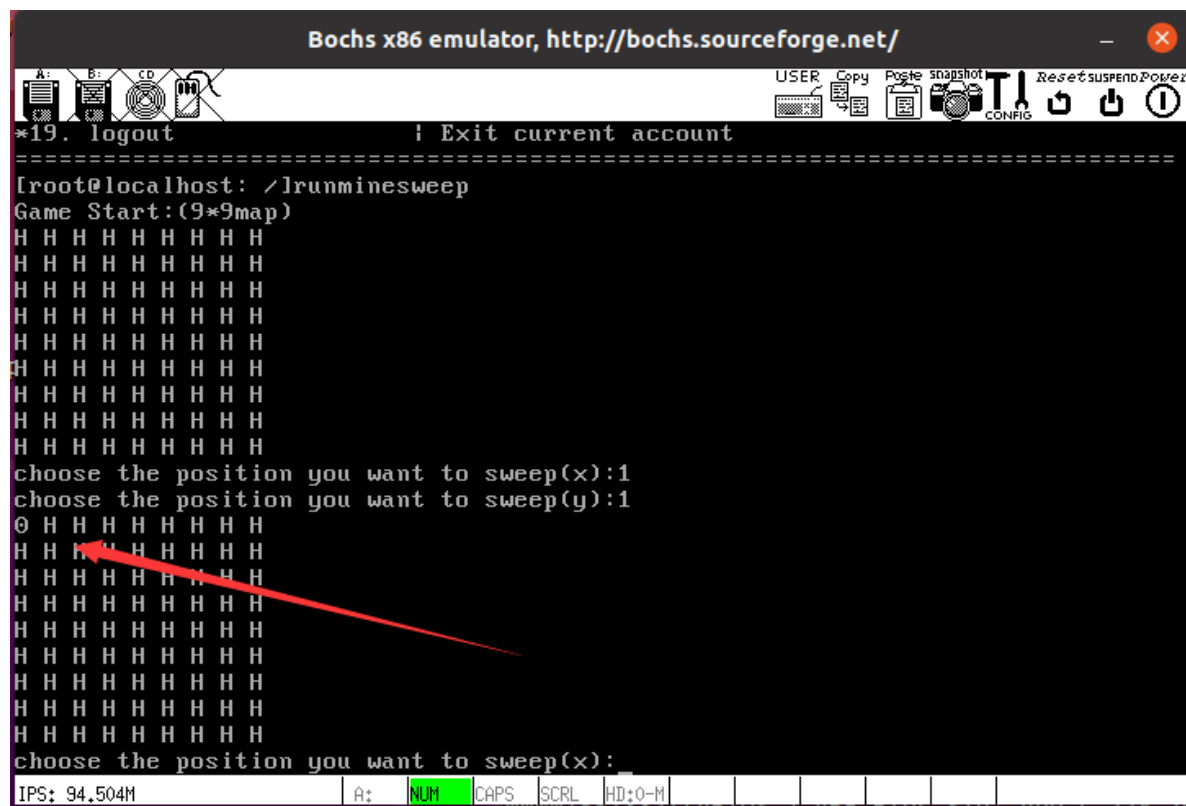
运行runminesweep命令进入扫雷界面

进入扫雷

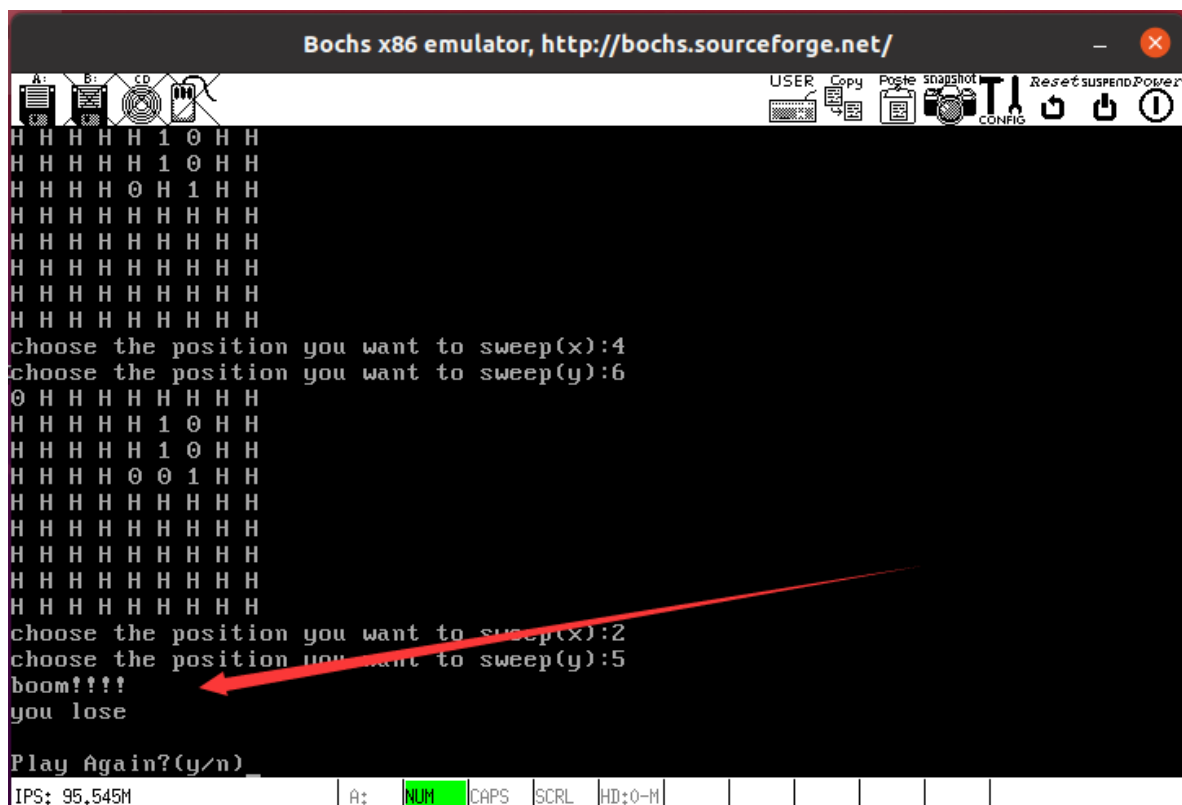
依次输入横纵坐标指定位置



展示当前位置数字，表面周围还有几个雷



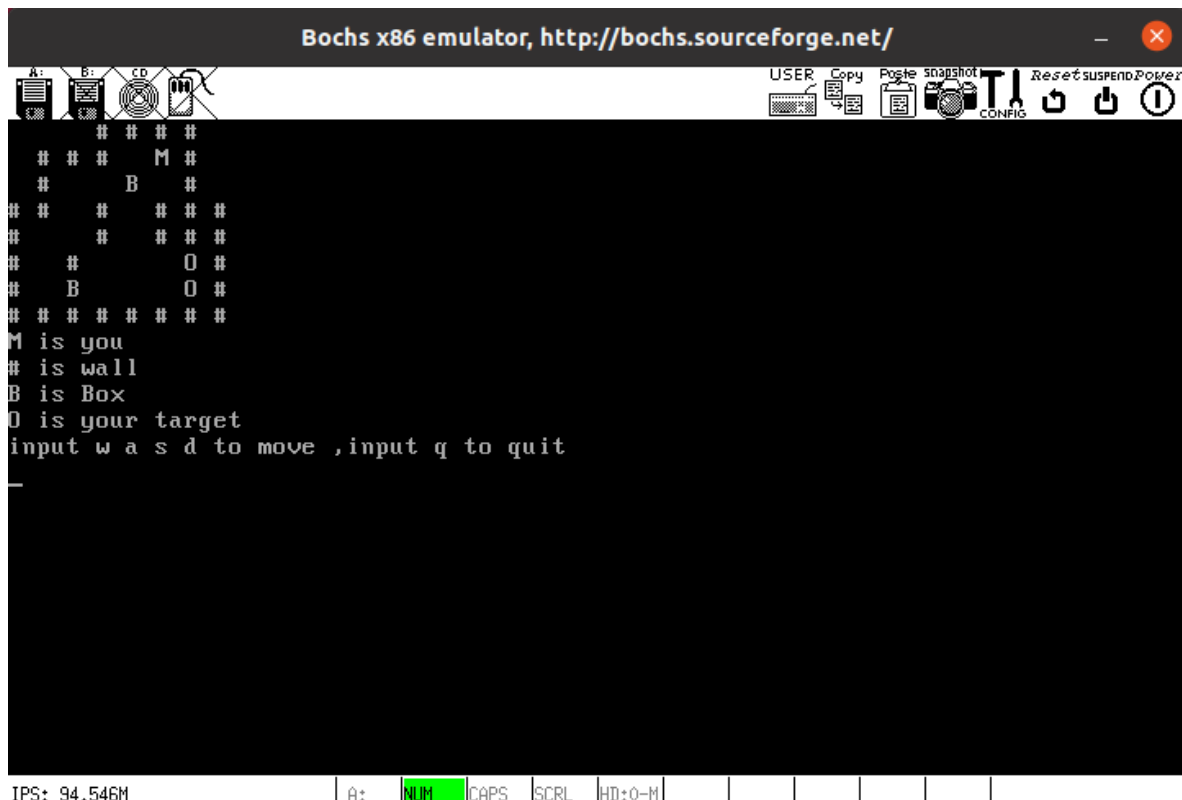
输出boom，表示游戏失败，输入y/n选择是否继续游戏



## 2.7.2 推箱子

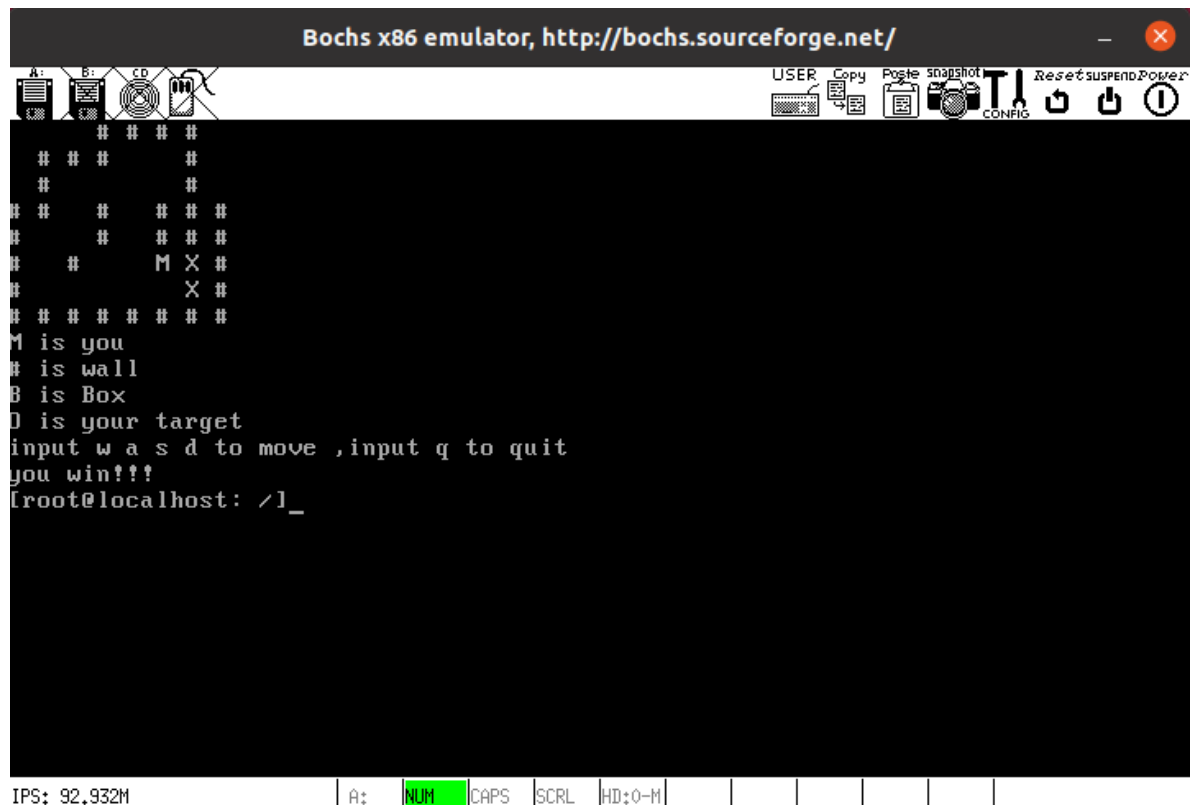
界面展示以及操作说明：

1. M表示玩家
2. #表示不可逾越的墙体
3. B表示要推的箱子
4. O表示将箱子推到目标
5. 使用w、a、s、d操作人物，q退出



当人物将箱子推到O处，变为X表示推到的目标位置

当两个箱子都到达了目标位置，则游戏胜利



### 3. 关键源码说明

#### 3.1 用户登录

主要使用以下函数进行密码验证：

```
if (login == 0) {
    int res = login11();
    while (res != 1) {
        res = login11();
    }
    login = 1;
    res = 0;
}
```

#### 3.2 控制台

读入命令后进行解析，处理路径和命令，并对比命令字符串调用对应的函数执行。

```
while (1)//不断循环的操作
{
    if (login == 0) {
        int res = login11();
        while (res != 1) {
            res = login11();
        }
        login = 1;
        res = 0;
    }
    printf("[root@localhost: %s]", currentFolder); // 打印当前路径
```

```
int r = read(fd_stdin, readBuffer, 512); //读入 r为大小  
readBuffer[r] = '\0';  
  
//解析命令  
int pos = 0;  
while (readBuffer[pos] != ' ' && readBuffer[pos] != '\0') // 读取指令这一字符串，以空格为分隔符  
{  
    command[pos] = readBuffer[pos];  
    pos++;  
}  
command[pos] = '\0'; //末尾添加一个0作为结束符  
if (readBuffer[pos] != '\0') // 指令还未结束  
{  
    pos++;  
    int len = pos;  
    while (readBuffer[pos] != ' ' && readBuffer[pos] != '\0') // 读取操作文件名  
    {  
        filename[pos - len] = readBuffer[pos];  
        pos++;  
    }  
    filename[pos - len] = '\0';  
}  
  
if (strcmp(command, "process") == 0) //开始比较指令内容  
{  
    ProcessManage(pidPoul);  
}  
else if (strcmp(command, "clear") == 0)  
{  
    clear();  
    printf("=====\n");  
    printf("          _   _     __\n\\\ |__   ____ \\\n\n");  
    printf("         /      \\ /  _ \\ /  _ \\|___ \| |\n\\\ /\n");  
    printf("       |  | ( < > | < >) \\_\\_\n | />\n< \n");  
    printf("      |_| /\\_____/ \\_____|___\n/_//_/\n\\\n");  
    printf("              \\\n                  \\\n\\\n");  
    printf("=====\\n");  
}  
else if (strcmp(command, "help") == 0)  
{  
    help();  
}  
else if (strcmp(command, "filemanage") == 0)  
{  
    printf("File Manager has already run on CONSOLE-1 ! \n");  
    continue;  
}  
else if (strcmp(command, "ls") == 0)
```



```

{
    ls(currentFolder);
}
else if (strcmp(command, "find") == 0)
{
    find_file(filename);
}
else if (strcmp(command, "touch") == 0) // 创建文件
{
    if(strcmp(currentFolder,"/bin/")==0){
        printf("can't create dir in the bin\n");
    }
    else{
        new_file(currentFolder, filename);
    }
}
else if (strcmp(command, "rmfile") == 0) // 删除文件
{
    delete_file(currentFolder, filename);
}
else if (strcmp(command, "rsfile") == 0)
{
    restore_file(currentFolder, filename);
}
else if (strcmp(command, "rmdir") == 0) // 删除目录
{
    delete_node(currentFolder, filename);
}
else if (strcmp(command, "cat") == 0) // 打印文件内容
{
    read_file(currentFolder, filename);
}
else if (strcmp(command, "vi") == 0) // 写文件
{
    if(strcmp(currentFolder,"/bin/")==0){
        printf("can't write file in the bin\n");
    }
    else{
        write_file(currentFolder, filename);
    }
}
else if (strcmp(command, "mkdir") == 0) // 创建目录
{
    new_dir(currentFolder, filename);
}
else if(strcmp(command,"clearbin")==0){
    clear_bin();
}
else if (strcmp(command, "cd") == 0)
{
    go_dir(currentFolder, filename);
}
else if (strcmp(command, "runminesweep") == 0)
{
    Minesweeper1(fd_stdin);
}
else if (strcmp(command, "runmovebox") == 0)
{

```

```

        MoveBox(fd_stdin);
    }
    else if (strcmp(command, "password") == 0)
    {
        password();
    }
    else if (strcmp(command, "getpid") == 0) //进程相关
    {
        printf(asm_strcat(getpid(), "\n"));
        printi(getpid());
        printf("\n");
        printi(new_getpid());
        printf("\n");
    }
    else if (strcmp(command, "fork") == 0)
    {
        int pid = fork();
    }
    else if (strcmp(command, "kill") == 0)
    {
        int pid = filename[0] - 48;
        kill_proc(pid, pidPoul);
        ProcessManage(pidPoul);
    }
    else if (strcmp(command, "block") == 0)
    {
        int pid = filename[0] - 48;
        block_proc(pid, pidPoul);
        ProcessManage(pidPoul);
    }
    else if (strcmp(command, "unlock") == 0)
    {
        int pid = filename[0] - 48;
        unblock_proc(pid, pidPoul);
        ProcessManage(pidPoul);
    }
    else if (strcmp(command, "logout") == 0)
    {
        clear();
        login = 0;
    }
    else
        printf("Command not found, please check!\n");
}

```

### 3.3 文件系统

文件系统功能	调用的函数
生成径	PUBLIC void convert(char* dest, char* path, char* file)
创建文件	void new_file(char* path, char* file)
创建目录	void new_dir(char* path, char* file)
删除目录	void delete_node(char* path, char* file)
删除文件	void delete_file(char* path, char* file)
回收垃圾	void restore_file(char* path, char* file)
前往目录	void go_dir(char* path, char* file)
读取目录	PUBLIC int ls(char* pathName)
写文件	void write_file(char* path, char* file)
读文件	void read_file(char* path, char* file)
更改密码	void password()
寻找文件	void find_file(char* filename)

### 3.3.1 文件系统的创建与删除

#### 3.3.1.1创建文件函数

```
void new_file(char* path, char* file)
{
    //用于存放字符串的末尾
    char buf[1] = {0};

    //用于存储绝对路径
    char abs_path[512];
    //将地址和文件名转换为绝对路径
    convert(abs_path, path, file);

    //新建文件
    int fd = open(abs_path, O_CREAT | O_RDWR);

    //如果创建失败输出失败信息
    if (fd == -1){printf("couldn't create a new file named %s\n", file);return;}

    //在文件的末尾写0
    write(fd, buf, 1);

    //关闭文件
    close(fd);

    if(strcmp("/bin/",path)==0)printf("File has moved to the bin\n");
}
```

```
}
```

### 3.3.1.2删除文件函数

```
void delete_file(char* path, char* file)
{
    //用于存储绝对路径
    char abs_path[512];

    //转换为绝对路径
    convert(abs_path, path, file);
    if(strcmp(abs_path, "/password") == 0)
    {
        printf("you cant delete this file\n");
        return;
    }
    if(strcmp(abs_path, "/bin")==0){
        printf("you cant delete the bin\n");
        return;
    }
    if(strcmp(path, "/bin/") == 0){
        //删除文件，并根据删除是否成功输出对应信息
        if (unlink(abs_path) == 0){printf("successfully deleted!\n");}
        else{printf("couldnt delete this file\n");}
    }
    else{

        char content[512]; //content of the file
        //打开文件并返回文件描述符
        int fd = open(abs_path, O_RDWR);
        //如果文件不存在输出错误信息
        if (fd == -1) {printf("delete error\n");return;}
        //向buf中读取文件内容
        int m = read(fd, content, 512);
        //如果文件读取失败输出错误信息
        if (m == -1){
            printf("couldnt restore the content of the file\n");
            //关闭文件
            close(fd);
            return;
        }
        close(fd);

        //删除文件
        convert(abs_path, path, file);
        if (unlink(abs_path) == 0){printf("successfully deleted!\n");}
        else printf("couldnt delete this file\n");

        //在回收站放置该文件
        new_file("/bin/", file);

        //重写文件
        char bin_path[512];
        convert(bin_path, "/bin/", file);
        int _fd = open(bin_path, O_RDWR);
        if (_fd == -1){printf("delete error");return;}
```

```

        write(_fd, content, 512);
        close(_fd);

    }
}

```

### 3.3.1.3创建目录函数

```

void new_dir(char* path, char* file)
{
    //存储绝对路径
    char abs_path[512];
    //改变为绝对路径
    convert(abs_path, path, file);

    //打开绝对路径下的文件
    int fd = open(abs_path, O_RDWR);

    //如果不能正常打开输出错误信息
    if (fd != -1) { printf("Failed to create a new directory with name %s\n",
file);return;    }

    //在对应位置创建新目录
    mkdir(abs_path);
}

```

### 3.3.1.4删除目录函数

```

void delete_node(char* path, char* file)
{
    //用于存储绝对路径
    char abs_path[512];

    //转换为绝对路径
    convert(abs_path, path, file);
    if(strcmp(abs_path, "/password") == 0)
    {
        printf("you cant delete this directory\n");
        return;
    }
    if(strcmp(abs_path, "/bin")==0){
        printf("you cant delete the bin\n");
        return;
    }
    //删除文件，并根据删除是否成功输出对应信息
    if (unlink(abs_path) == 0)printf("successfully deleted!\n");

    else printf("couldnt delete this file\n");

}

```

### 3.3.1.5恢复文件函数

```

void restore_file(char* path, char* file)

```

```

{

    //用于存储绝对路径
    char abs_path[512];

    //转换为绝对路径
    convert(abs_path, path, file);

    if(strcmp(path, "/bin/") != 0){
        //删除文件，并根据删除是否成功输出对应信息
        printf("you cannot restore the file out of the bin!\n");
        return;
    }
    else{

        char content[512]; //content of the file
        //打开文件并返回文件描述符
        int fd = open(abs_path, O_RDWR);
        //如果文件不存在输出错误信息
        if (fd == -1) {printf("restore error\n"); return; }
        //向buf中读取文件内容
        int m = read(fd, content, 512);
        //如果文件读取失败输出错误信息
        if (m == -1){
            printf("couldnt restore the content of the file\n");
            //关闭文件
            close(fd);
            return;
        }
        close(fd);

        //删除回收站中文件备份
        convert(abs_path, path, file);
        if (unlink(abs_path) == 0);
        else printf("couldnt restore this file\n");

        //在根目录恢复原文件
        new_file("/", file);

        char old_path[512];
        convert(old_path, "/", file);
        int _fd = open(old_path, O_RDWR);
        if (_fd == -1){printf("delete error"); return;}
        write(_fd, content, 512);
        close(_fd);

    }
}

```

### 3.3.1.6 清空文件夹函数

```

void clear_bin()
{
    //删除原回收站目录并新建新的bin目录替代
    char abs_path[512];
    convert(abs_path, "/", "bin");
    if (unlink(abs_path) == 0){
        printf("successfully clear the bin!\n");
        convert(abs_path, "/", "bin");
        int fd = open(abs_path, O_RDWR);
        if(fd!=-1)mkdir("/bin");
    }
    else printf("couldnt delete this directory\n");
}

```

### 3.3.2 文件系统的访问与读写

#### 3.3.2.1 合并路径函数

```

PUBLIC void convert(char* dest, char* path, char* file)
{
    int i=0, j=0,m=0;
    while (path[i]) dest[j++] = path[i++];

    while (file[m]) dest[j++] = file[m++]; // 写入文件名

    dest[j] = 0; // 结束符
}

```

由于在文件系统中路径和文件名时分开的，但是在对文件进行访问时，使用的时文件的绝对路径。因此，在使用前应该将文件的目录路径和文件名转换为绝对路径方便进行访问

#### 3.3.2.2 访问目录

总体上将访问目录的情况分成两种。一种时`cd+ 文件名`，另一种是`cd ..`，并对这两种情况分别进行编写。具体上还需要对于特殊目录如`protect`等进行一定程度的保护，防止对于保护目录的直接访问（具体可看注释）

```

void go_dir(char* path, char* file)
{
    int flag = 0;
    //保存新路径，因为可能出现倒退的情况，因此需要new_path记录上一级的目录
    char new_path[512] = {0};

    //对于指令cd..在DOS系统中一般用于返回上一级
    if (file[0] == '.' && file[1] == '.')
    {
        flag = 1;

        //在路径字符串中的字符位置，用于遍历
        int pos_path = 0;

        //在新的路径字符串的字符位置，用于遍历
        int pos_new = 0;
    }
}

```

```

//i用于temp的计数
int i = 0;

//temp作为中间变量保存过程中的文件目录名
char temp[128] = {0};

//如果没有到达字符串的结尾
while (path[pos_path] != 0)
{

    //未到达'/'时
    if (path[pos_path] != '/')
    {
        //使用temp保存路径
        temp[i] = path[pos_path];

        i++;

        pos_path++;
    }

    //如果到达'/'时
    else
    {
        //递增计数
        pos_path++;

        //到达结尾时停止
        if (path[pos_path] == 0)break;

        //如果不是结尾
        else
        {
            //对temp进行初始化
            temp[i] = '/';

            //末尾置零
            temp[i + 1] = 0;

            i = 0;

            //将temp中的全部复制到newPath中再把temp清除
            while (temp[i] != 0)
            {

                new_path[pos_new] = temp[i];

                temp[i] = 0;

                pos_new++;

                i++;
            }
            i = 0;
        }
    }
}

```



```

}

//存储绝对路径
char abs_path[512];

//存储文件名称
char file_name[512];

int pos = 0;

//将文件名称复制过来
while (file[pos] != 0){file_name[pos] = file[pos];pos++; }

//给文件末尾添加终止符号
file_name[pos] = '/';
file_name[pos + 1] = 0;

//返回上一级目录
if (flag == 1) {file_name[0] = 0;convert(abs_path, new_path, file_name); }

//进入下一级目录
else convert(abs_path, path, file_name);

if(strcmp(abs_path, "/protect/")==0)
{
    char buf1[512];
    int fd = open("/password", O_RDWR);
    read(fd, buf1, 512);

    char tty[] = "/dev/tty0";
    int input = open(tty, O_RDWR);

    char write_buf[4096];

    //length为读取的字符串长度,将输入读入写缓冲区中

    printf("please enter your password\n");
    int length = read(input, write_buf, 4096);

    //将末尾置0, 保证字符串正常结束
    write_buf[length] = 0;

    if(strcmp(buf1, write_buf) != 0){printf("wrong password\n");return;}

}

//打开文件
int fd = open(abs_path, O_RDWR);

//如果打开失败输出错误信息
if (fd == -1) printf("%s is not a directory!\n", abs_path);

//将当前的绝对路径复制给所在目录的路径
else memcpy(path, abs_path, 512);
}

```

### 3.3.2.3 展示目录内容

此处主要是对于书中代码的模仿，通过发送消息的方式实现目录内容的展示

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

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

    send_recv(BOTH, TASK_FS, &msg);

    return msg.RETVAL;
}
```

### 3.3.2.4 写文件

通过虚拟终端tty将内容输入到系统中，再使用书中写好的代码将内容写入到对应文件中

```
void write_file(char* path, char* file)
{
    //保存绝对路径
    char abs_path[512];

    //转换为绝对路径
    convert(abs_path, path, file);

    //根据绝对路径进行打开
    int fd = open(abs_path, O_RDWR);

    //如果打开失败输出错误信息
    if (fd == -1){printf("couldnt open file: %s!\n", file);return; }

    //指定读写需要的TTY
    char tty[] = "/dev/tty0";

    //打开dev_tty0，由于是在文件系统开始就已经创建了这个文件，因此这个文件一定存在不需要判断是否存在
    int input = open(tty, O_RDWR);

    char write_buf[4096];

    //length为读取的字符串长度，将输入读入写缓冲区中
    int length = read(input, write_buf, 4096);

    //将末尾置0，保证字符串正常结束
    write_buf[length] = 0;

    //将写缓冲区中的内容写入fd对应的文件
    write(fd, write_buf, length + 1);

    //关闭文件
}
```

```
close(fd);  
}
```

### 3.3.2.5 读文件

使用convert函数求出文件的绝对路径之后，调用read函数将内容读出到缓冲区，再将缓冲区中内容使用printf进行输出

```
void read_file(char* path, char* file)  
{  
  
    //保存绝对路径  
    char abs_path[512];  
    char buf[512];  
  
    //转换为绝对路径  
    convert(abs_path, path, file);  
  
    //打开文件并返回文件描述符  
    int fd = open(abs_path, O_RDWR);  
  
    //如果文件不存在输出错误信息  
    if (fd == -1) {printf("couldnt read %s!\n", file);return; }  
  
    if(strcmp(abs_path, "/password") == 0)  
    {  
  
        printf("you cant read this file\n");  
        return;  
    }  
  
    //向buf中读取文件内容  
    int m = read(fd, buf, 512);  
  
    //如果文件读取失败输出错误信息  
    if (m == -1)    {  
  
        printf("couldnt read this file\n");  
  
        //关闭文件  
        close(fd);  
  
        return;  
    }  
  
    //输出文件内容  
    printf("%s\n", buf);  
  
    //关闭文件  
    close(fd);  
}
```

### 3.3.3 文件的保护

这段代码写于go\_dir函数中，用以再访问protect的时候使用密码加以限制

```
char buf1[512];
    int fd = open("/password", O_RDWR);
    read(fd, buf1, 512);

    char tty[] = "/dev_tty0";
    int input = open(tty, O_RDWR);

    char write_buf[4096];

    //length为读取的字符串长度,将输入读入写缓冲区中

    printf("please enter your password\n");
    int length = read(input, write_buf, 4096);

    //将末尾置0,保证字符串正常结束
    write_buf[length] = 0;

    if(strcmp(buf1, write_buf) != 0){printf("wrong password\n");return;}
```

### 3.3.4 文件检索

为了节约内存的开销防止系统崩溃，使用栈（队列实现也可以）来存储将要访问的子文件夹来实现直接搜索全部的文件

```
PUBLIC int do_find(const char *_FILENAME)
{
    printf("\ninode      location      \n");
    printf("===== \n");

    PUSH(stack, "/");
    while(top!=0)
    {
        char pathName[128];
        char* temp=POP(stack).address;
        memcpy(pathName,temp,128) ;

        // 取得message中的信息, 详见lib/ls.c
        int name_len=strlen(pathName);

        assert(name_len < 128); // 路径名称长度不得超过最大长度

        pathName[name_len] = 0;

        int i, j;

        //struct inode * dir_inode = root_inode;
        struct inode * dir_inode; // 需要令它指向当前的目录节点
        char fileName[20];
        strip_path(fileName, pathName,&dir_inode);

        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;
int m = 0;

struct dir_entry * pde;
struct inode* new_inode; // 指向每一个被遍历到的节点

for (i = 0; i < nr_dir_blks; i++)
{
    RD_SECT(dir_inode->i_dev, dir_blk0_nr + i);

    pde = (struct dir_entry *)fsbuf;
    for (j = 0; j < SECTOR_SIZE / DIR_ENTRY_SIZE; j++, pde++)
    {
        if (pde->inode_nr == 0)
            continue;
        if (pde->type == 'd')
        {
            char t[128];
            char tempPath[128];
            char newPath[128];
            memcpy(t, pathName, 128);
            int len=strlen(t);
            t[len]=0;
            memcpy(tempPath, strcat(t, pde->name), 128);
            len=strlen(tempPath);
            tempPath[len]=0;
            memcpy(newPath, strcat(tempPath, "/"), 128);
            len=strlen(newPath);
            newPath[len]=0;

            PUSH(stack, newPath);
        }
        else
        {
            if(strcmp(pde->name, _FILENAME)==0){
                printf(" %2d %s\n", pde->inode_nr, pathName, pde-
>name);
            }
        }
        if (++m >= nr_dir_entries)
        {
            printf("\n");
            break;
        }
    }
    if (m > nr_dir_entries) //[> all entries have been iterated <]
        break;
}
}
printf("=====\n");
return 0;
}

```

## 3.4 进程模块

### 3.4.1 进入进程模块

```

void ProcessManage(int* pidPoul)
{
    int i;
    printf("-----\n");
    //进程号, 进程名, 优先级, 是否是系统进程, 是否在运行
    printf("      PID      |   name      | spriority   | running?\n");
    printf("-----\n");
    for ( i = 0 ; i < NR_TASKS + NR_PROCS ; i++ )//逐个遍历
    {
        if (proc_table[i].p_flags != FREE_SLOT)
        {
            if (pidPoul[i] == 1)
            {
                continue;
            }
            else if (pidPoul[i] == 2)
            {
                printf("      %d      %s      %d\n", proc_table[i].pid, proc_table[i].name, proc_table[i].priority);
            }
            else if (pidPoul[i] == 0)
            {
                printf("      %d      %s      %d\n", proc_table[i].pid, proc_table[i].name, proc_table[i].priority, proc_table[i].p_flags == FREE_SLOT ? "NO" : "YES");
            }
        }
    }
    printf("-----\n");

    printf("=====\n");
    printf("*                      Process Command List:\n");
    printf("*                      Please Enter the command!\n");
    printf("=====\n");
    printf("*1. kill                [pid]| Kill a process by its pid\n");
    printf("*2. block                [pid]| Block a process by its pid\n");
    printf("*3. unblock              [pid]| Unblock a process by its pid\n");
    printf("*4. fork                  | Fork a process\n");

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

```

### 3.4.2 创建子进程

```

PUBLIC int fork()
{
    MESSAGE msg;
    msg.type = FORK;

    send_recv(BOTH, TASK_MM, &msg);
    assert(msg.type == SYSCALL_RET);
    assert(msg.RETVAL == 0);

    return msg.PID;
}

```

### 3.4.3 阻塞进程

```

void block_proc(int pid, int*pidPoul)
{
    if (pid >= NR_TASKS + NR_PROCS && pid < 0)
    {
        printf("                warning:This pid is out of range\n");
    }
    else if (pidPoul[pid] == 1)
    {
        printf("                warning:This pid has been killed\n");
    }
    else if (pidPoul[pid] == 2)
    {
        printf("                warning:This pid has been blocked\n");
    }
    else if (pidPoul[pid] == 0)
    {
        pidPoul[pid] = 2;
        printf("                Process %s has been blocked\n", proc_table[pid].name);
    }
}

```

### 3.4.4 恢复进程

```

void unblock_proc(int pid, int*pidPoul)
{
    if (pid >= NR_TASKS + NR_PROCS && pid < 0)
    {
        printf("                warning:This pid is out of range\n");
    }
    else if (pidPoul[pid] == 1)
    {
        printf("                warning:This pid has been killed\n");
    }
    else if (pidPoul[pid] == 0)
    {

```

```

        printf("                warnning:This pid is still running\n");
    }
    else if (pidPoul[pid] == 2)
    {
        pidPoul[pid] = 0;
        printf("                Process %s has been unblocked\n", proc_table[pid].name);
    }
}

```

### 3.4.5 结束进程

```

void kill_proc(int pid, int*pidPoul)
{
    if (pid >= NR_TASKS + NR_PROCS && pid < 0)
    {
        printf("                warnning:This pid is out of range\n");
    }
    else if (pidPoul[pid] == 1)
    {
        printf("                warnning:This pid has been killed\n");
    }
    else if (pidPoul[pid] == 0 || pidPoul[pid] == 2)
    {
        pidPoul[pid] = 1;
        printf("                Process %s has been killed\n", proc_table[pid].name);
    }
}
}
}

```

## 3.5 多终端

二号终端所运行的内容

```

void TestB()
{
    char tty_name[] = "/dev_tty1";

    int fd_stdin = open(tty_name, O_RDWR);
    assert(fd_stdin == 0);
    int fd_stdout = open(tty_name, O_RDWR);
    assert(fd_stdout == 1);
    char rdbuf[128];
    while (1)
    {
        printf("$ ");
        int r = read(fd_stdin, rdbuf, 70);
    }
    assert(0); /* never arrive here */
}

```



## 3.6 游戏

### 3.6.1 扫雷

```
void add(int x, int y); //获得每一个坐标附近雷的数量
void Init(); //初始化
void PrintMap();
int Judge(int x, int y); //判断是否爆炸
int UserInput(int fd_stdin); //用户输入
void Minesweeper1(int fd_stdin) { //主函数
    count = 100;
    char buf[80] = { 0 };
    char IsFirst = 0;
    int IsFinish = FALSE;
    while (!IsFinish)
    {
        printf("Game Start:(9*9map)\n");
        Minesweeper();
        do {
            IsFinish = UserInput(fd_stdin);
            if (!IsFinish) {
                PrintMap();
            }
        } while (!IsFinish);
        if (IsFinish)
        {
            if (count > 10) {
                printf("boom!!!!\nyou lose\n\n");
            }
            else {
                printf("you get all the mine!\n");
            }

            printf("Play Again?(y/n)");
            char cResult;
            read(fd_stdin, buf, 2);
            cResult = buf[0];
            printf("%c", cResult);
            if (cResult == 'y')
            {
                clear();
                IsFinish = FALSE;
                count = 100;
            }
            else
            {
                clear();
            }
        }
    }
}
```

### 3.6.2 推箱子

```
void initData(); //初始化数据
void PrintM(); //打印地图
```

```
void Move(int dir); //根据输入移动小人
void MoveBox(int fd_stdin) {
    char buf[80] = { 0 };
    char direction;
    initData();
    while (1) {
        clear();
        PrintM();

        if (!boxs) {
            break;
        }
        read(fd_stdin, buf, 2);
        direction = buf[0];
        switch (direction) {
            case 'w':
                Move(1);
                break;
            case 'a':
                Move(2);
                break;
            case 's':
                Move(3);
                break;
            case 'd':
                Move(4);
                break;
            case 'q':
                return;
                break;
        }
    }
    printf("you win!!!\n");
}
```