|  |
| --- |
|  |
| 操作系统课程设计 |
|  |

一.项目设计

本次操作系统课程设计，我们小组选择对orange源码进行修改，对包括进程管理与文件管理部分进行修改与加强，并在已有的orange操作系统之上，实现一些列交互友好的用户级程序（游戏）。具体实现要点如下：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型与说明 | 难度 |
| ProcessManage | 进程管理模块，基于书本进行修改与重新实现 | B |
| FileSystem | 文件管理模块，基于书本的文件管理模块进行修改 | B |
| GuessNumber | 用户级应用，数字猜测小游戏 | D |
| Compile | 用户级应用，实现了一个简单的编译系统，对简单的类C输入可以进行编译计算 | D |
| Maze | 用户级应用，一个简单的迷宫小游戏 | D |
| Snake | 用户级应用，一个小型的贪吃蛇小游戏 | D |
| WuziChess | 用户级应用，一个简单的五子棋小游戏，拥有AI对战功能 | D |

二.项目配置

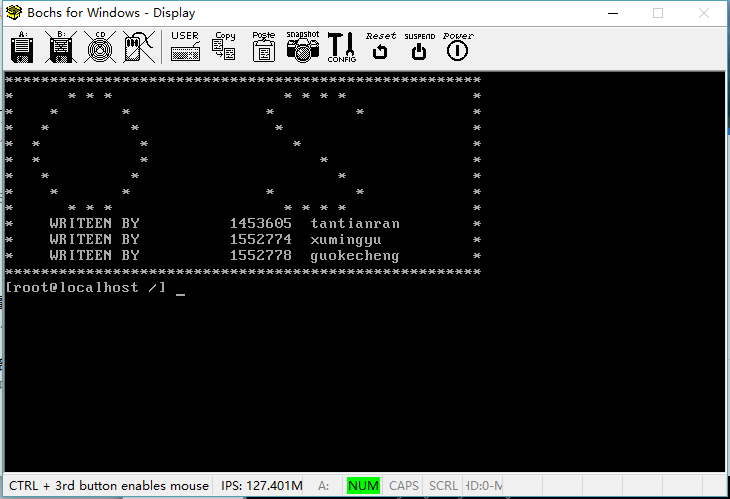
编写语言： C、汇编

开发环境： Windows ，Ubuntu下的Bochs

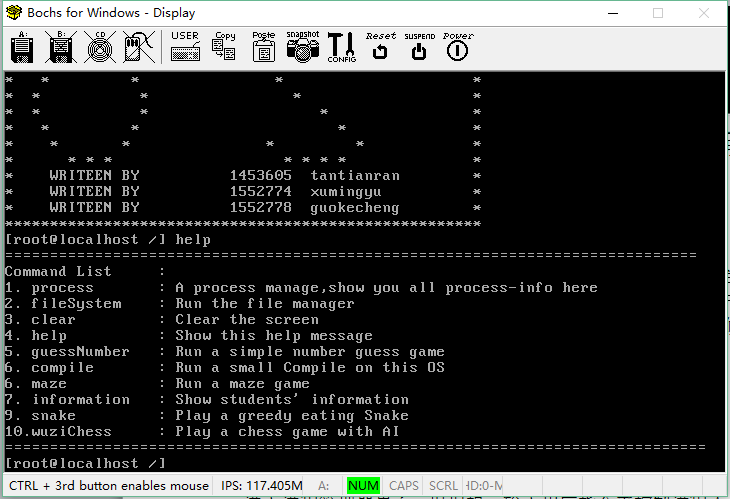
运行环境： Ubuntu、Windows

三.项目详解

运行bochsrc.bxrc；即可看到虚拟操作系统：



使用“help”命令即可看到所有指令，其后均有详细说明。



ProcessManage：

使用processManage指令进入ProcessManage界面：

设计思路：增加一个全局变量schedule\_flag来记录当前选择的进程调度方法，实现轮转法和优先权法的转换；在PCB中新加入p\_runable来实现进程的暂停和运行；通过用户的指令相应修改这两个值及priority来实现进程管理器的功能。

源码位置：kernel/main.c

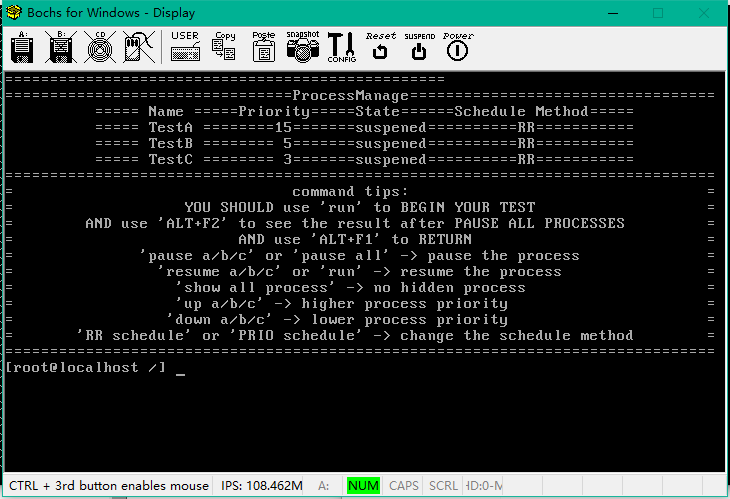
kernel/clock.c

include/sys/global.h

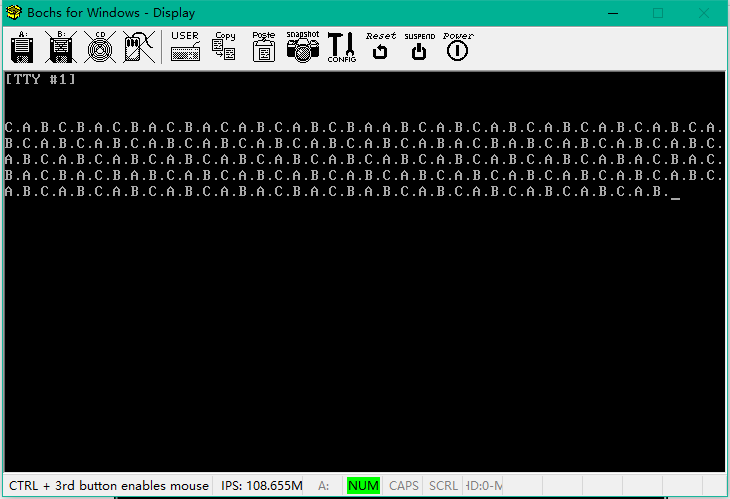
include/sys/proc.h

功能说明：

进入进程管理器界面，根据提示输入相应指令来控制进程A、B、C的运行。

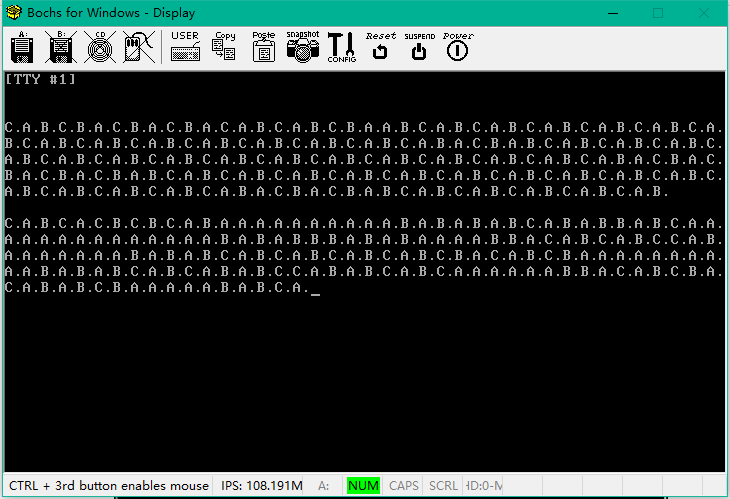


输入‘run’来控制三个进程开始打印，暂停后结果如下：



由于调度算法为轮转法，三种进程打印出的字符频率大致相同。

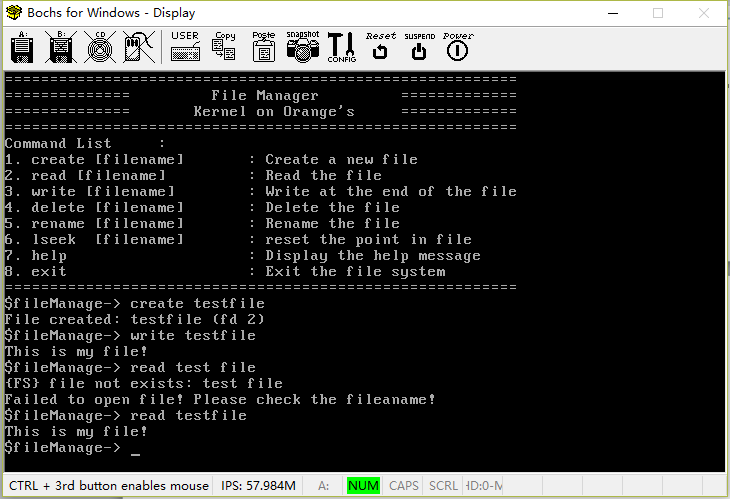
输入‘PRIO schedule’来切换到优先权调度，运行后结果如下：



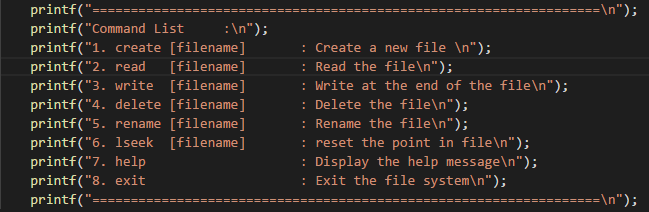
可以看到由于进程A、B、C有不同的优先级，切换后三种字符的打印频率由高到低为A、B、C。

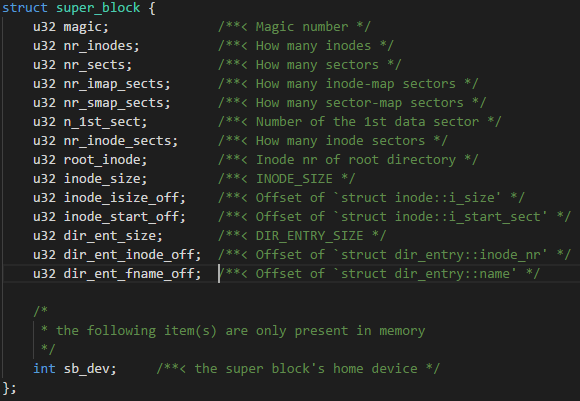
FileSystem：

使用fileSystem指令进入FileSystem界面：



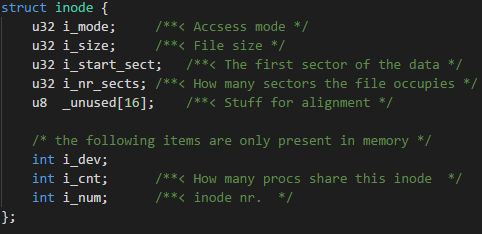
设计思路： 文件系统暂不支持文件夹，也就是说用来表示目录的特殊文件只有一个，即一个不支持文件夹的文件系统，即扁平文件系统,另外，整合了文件功能，使之支持文件的创立(create)，读写（read&write），删除（delete），重命名（rename）以及文件的重定位操作（lseek）。

 在include/sys/fs.h中,定义了super\_block(超级块)：



标明了文件系统的标识，文件系统最多所能允许有多少个i-node，inode\_array占用了多少扇区，文件系统总共扇区数是多少，inode-map占用多少扇区，sector-map占用多少扇区，第一个数据扇区的扇区号是多少以及根目录区的i-node号是多少。

此外，对i-node也进行了定义：



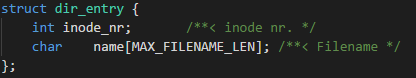
其中i\_start\_sect代表文件的起始扇区，i\_nr\_sects代表总扇区数，i\_size代表文件大小。使用静态分配的方式来存储文件，即事先为文件预留出一定的扇区数（i\_nr\_sects），以便让文件能够追加数据，i\_nr\_sects一旦确定就不再更改。这样做的有点很明确，那就是分配扇区的工作在建立文件时一次完成，从此再也不用额外分配或是释放扇区。但缺点亦很明确，即文件大小范围在文件建立以后就无法改变了，其中，i\_size范围在0到i\_nr\_sects512之间。

针对此问题，对文件系统进行小幅度修改，对每一个inode节点，添加一个u32结构数组：



当一个inode的i\_size大小用尽之后，开辟新的inode用于文件的存储，此u32[10]的数组便用于存储后续节点的i\_start\_sect，这样，文件的大小将会变为动态，最小为1个i\_size,最大为11个i\_size，但同时由于最后一个节点无法用完，可能一整个inode分配的空间会被浪费许多，造成浪费，因此，理论上i\_size节点越小越好，这样分配空间的利用率高，但是当i\_size过小后，需要频繁增删的文件则由于需要频繁的后续节点记录，会造成性能上的浪费，因此最终选择一个节点使用大小为512个字节。

最后的dir\_entry结构体：



它是存在于根目录文件中的数据结构，用于索引一个文件。拥有两个成员：i-node和文件名。根目录会是一个dir\_entry数组，用于索引文件系统中的所有文件。

源码位置： kernel/main.c

lib/write.c

lib/read.c

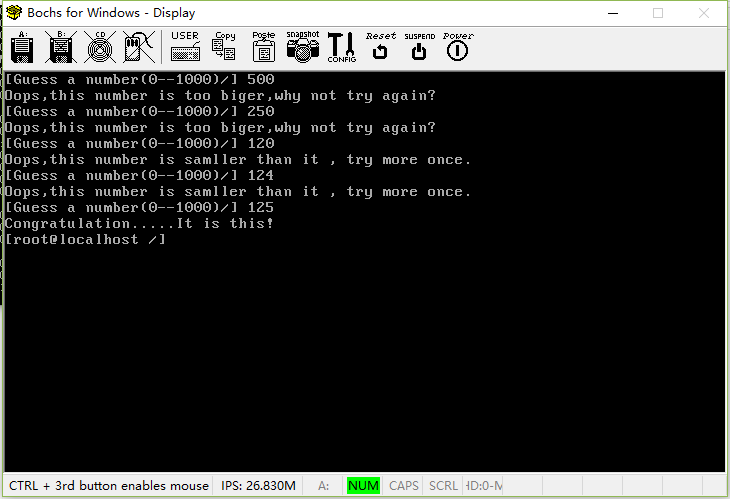
lib/open.c

lib/close.c

fs/link.c

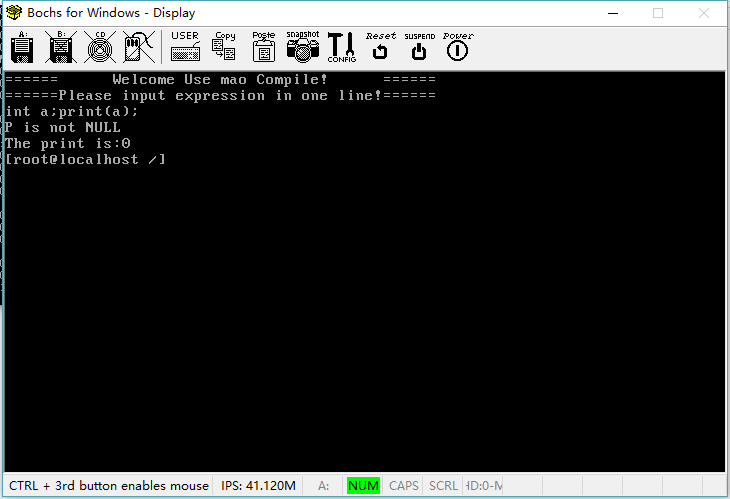
GuessNumber：

使用guessNumber指令进入GuessNumber游戏界面：相关指令详见控制台。



源码位置：kernel/main.c

Compile：

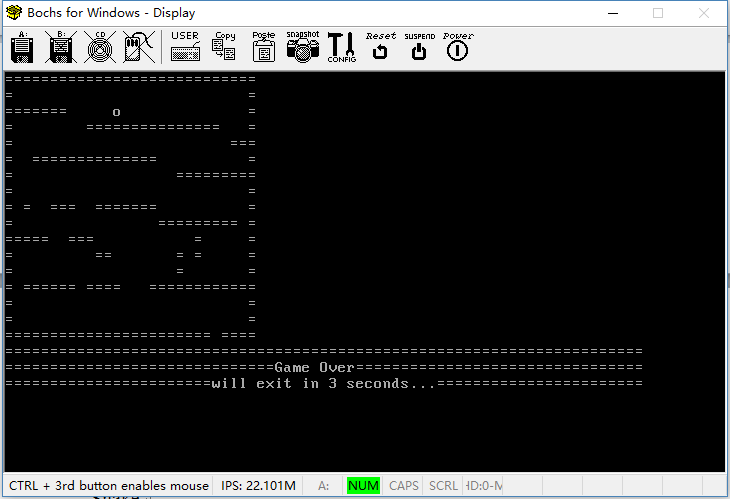
使用Compile指令进入Compile界面：

（注意：类C代码输入时，以输入回车为结束标志）代码需要放于一行内输入，支持int ,double类型，内部采用了逆波兰表达式对表达式进行计算与输出。

源码位置：kernel/main.c

Maze:

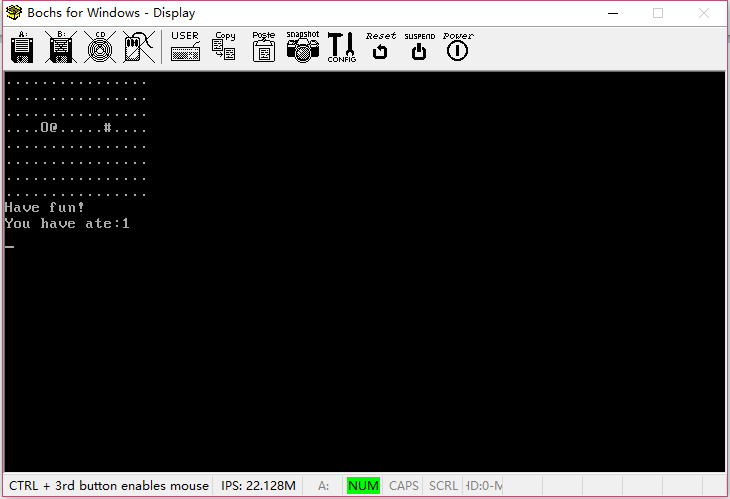
使用Maze指令进入Maze游戏界面：一个简单的迷宫游戏，使用WASD进行控制，当撞上墙壁时，游戏结束，不触碰任何墙壁而到达终点算游戏胜利。



源码位置：kernel/main.c

Snake:

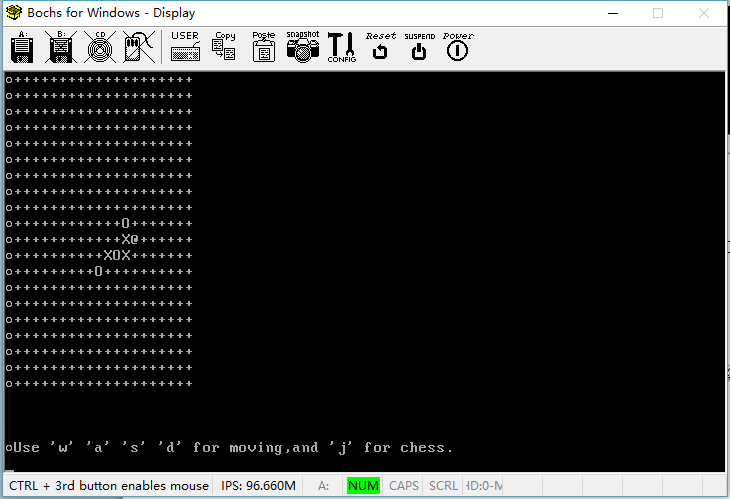
使用Snake指令进入Snake游戏界面：一个比较简单的贪吃蛇游戏，其中，@代表蛇头，0代表蛇身，#代表着食物，使用WASD进行操作，当蛇头触碰到墙头或是自身身体，游戏结束。



源码位置：kernel/main.c

wuziChess:

一个简单的五子棋小游戏，拥有与AI对战的功能，O代表白子，X代表黑子，@代表当前光标位置，详细操作请见控制台提示：



源码位置：kernel/main.c