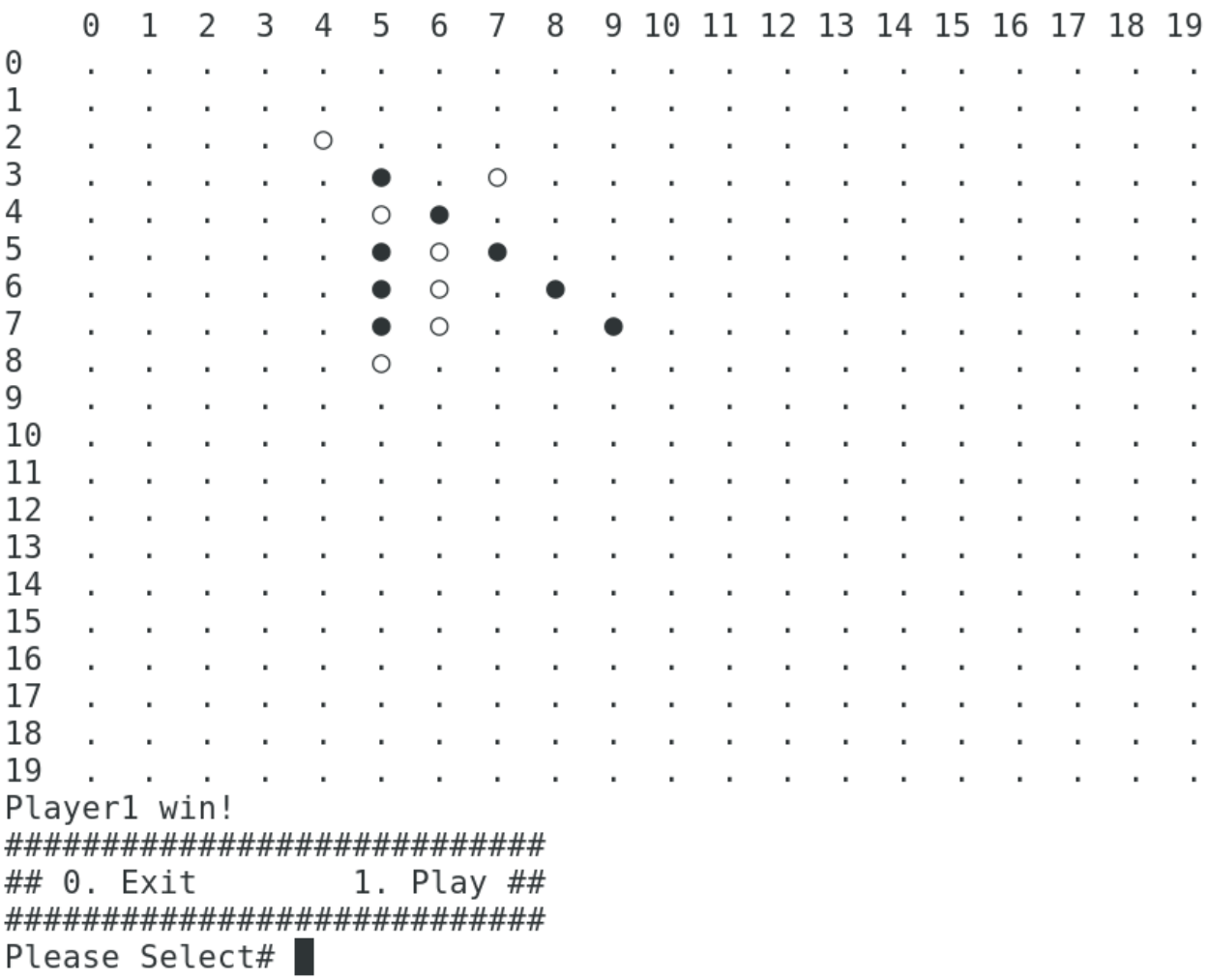
Linux小实战1-五子棋上

> - 巩固Linux环境指令操作
> - 巩固Linux中相关开发工具的基本使用
> - 练习在Linux环境当中进行C编程
> - 掌握五子棋的基本原理
> - 编写五子棋程序的上层基本调用框架

我们做的是什么样子?



五子棋原理

采用二维数组保存棋盘信息，棋盘上面的任何一个位置，里面可以放置三类信息

> - 空
> - 用户1的落子（黑子）
> - 用户2的落子（白子）

下棋就是在二维数组中找对应的空位置，进行落子

落完子之后下来就要考虑该落子位置是否有"五子连珠"，进而进行输赢判定，每一次走棋，多会有四种情况

> - 用户1赢
> - 用户2赢
> - 平局
> - 未出结果

其中，“未出结果”游戏要继续，其他三种情况，游戏不用继续

第一步. main构建游戏起始逻辑

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void Menu() //游戏菜单
{
  printf("#############################\n");
  printf("## 0. Exit        1. Play ##\n");
  printf("#############################\n");
  printf("Please Select# ");
}

int main()
{
  int quit = 0;
  int select = 0;
  while(!quit){
    Menu();
    scanf("%d", &select);
    switch(select){    //根据用户选择，执行合适的游戏逻辑代码
      case 0:
        quit = 1;
        break;
      case 1:
        Game();
        break;
      defalut:
        printf("Please Select Again!\n");
        break;
    }
  }
}
```

第二步. 构建游戏入口Game()函数

```
#define ROW 20        //数组行数，可以按照需求调整
#define COL 20        //数组列数，可以按照需求调整

#define PLAYER1  1    //玩家1编号，默认棋盘数据是0，玩家1落子，该位置被改成2
#define PLAYER2  2    //玩家2编号，默认棋盘数据是0，玩家1落子，该位置被改成2

#define NEXT 0        //游戏继续
#define PLAYER1_WIN 1 //玩家1赢了
```

```
#define PLAYER2_WIN 2 //玩家2赢了
#define DRAW 3        //平局

void Game()
{
  int board[ROW][COL];                  //采用ROW * COL型的二维数组，来进行游戏信息的保存
  memset(board, '\0', sizeof(board));  //默认棋盘数据都是0
  int result = NEXT;

  do{
    ShowBoard(board);                  //显示棋盘
    PlayerMove(board, PLAYER1);        //PLAYER1先走，也可以随机让一个人先走
    result = IsOver(board);            //判定游戏是否结束
    if(NEXT != result){
      break;
    }
    ShowBoard(board);
    PlayerMove(board, PLAYER2);
    result = IsOver(board);
    if(NEXT != result){
      break;
    }
  }while(1);

  ShowBoard(board);
  switch(result){
    case PLAYER1_WIN:
      printf("Player1 win!\n");
      break;
    case PLAYER2_WIN:
      printf("Player2 win!\n");
      break;
    case DRAW:
      printf("Player1 darw Player2!\n");
      break;
  }
}
```

第三步. 编写核心函数，定义全局游戏信息

```
//computer or player move postion,just for simple
//当前用户的落子位置，相对而言这样设计，能够简化游戏逻辑
int x = 0;
int y = 0;

void PlayerMove(int board[ROW][COL], int who)
{

}
int ChessCount(int board[ROW][COL], enum Dir d)
{
    return 0;
```

```
}
int IsOver(int board[ROW][COL])
{
    return NEXT;
}
void ShowBoard(int board[ROW][COL])
{

}
```

第四步. 编写Makefile，完成第一步项目构建

```
gobang:gobang.c
    gcc  -o $@ $^ -g -std=c99

.PHONY:clean
clean:
    rm -f gobang *.o
```

调试通过代码，准备工作完成