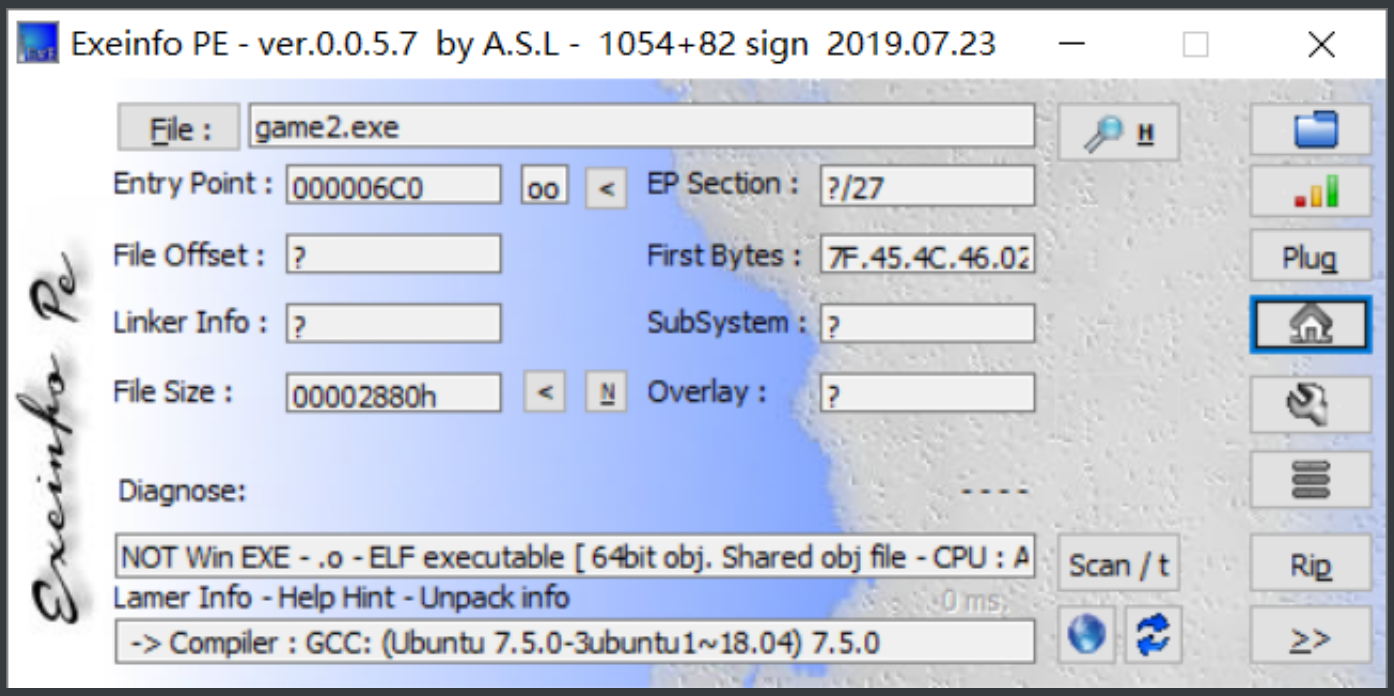
拿到附件，虽然是exe后缀但是打不开

名称	修改日期	类型	大小
game2.exe	2020/8/4 21:37	应用程序	11 KB

使用Exeinfo查看一下发现是ELF64文件



先简单运行一下程序

```
eliauk@Pluto: /mnt/f/palmer/题HebtuCTF2020
eliauk@Pluto:/mnt/f/palmer/题HebtuCTF2020$ ./game2.exe
Guojing wanted to go through the forest to get the treasure. And it won't be smooth sailing.
Can you help Guojing get the treasure?
From now on, she's going to follow your lead.
11111111111111111111
Following your instructions, she started exploring.
I don't think you got the right route....
eliauk@Pluto:/mnt/f/palmer/题HebtuCTF2020$
```

IDA64位载入 来到main函数 F5查看伪代码

(经过变量和函数重命名并加注释后的伪代码)

```
IDA View-A   Pseudocode-A   Hex View-1   Structures   Enums   Im
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  __int64 result; // rax
  FILE *v4; // rdi
  signed int i; // [rsp+4h] [rbp-1Ch]
```

```
int v6; // [rsp+8h] [rbp-18h]
int v7; // [rsp+10h] [rbp-10h]
int v8; // [rsp+14h] [rbp-Ch]

v6 = 0;
v8 = 0;
puts("Guojing wanted to go through the forest to get the treasure. And it won't be smooth sailing.");
puts("Can you help Guojing get the treasure?");
puts("From now on, she's going to follow your lead.");
__isoc99_scanf("%s", user_input);
puts("Following your instructions, she started exploring.");
if ( (unsigned int)strlen(user_input) == 32 ) // 判断长度是否等于32，若不等于直接失败
{
  for ( i = 0; i <= 31; ++i )
  {
    sleep(1u);
    v7 = v8;                                // 动调得到v7为保存当前循环的上上次按键
    v8 = v6;                                // v8保存当前循环的上次按键
    printf(&format);
    v4 = stdout;
    fflush(stdout);                         // 刷新输出缓冲区，打印实心方块
    v6 = user_input[i];                     // v6挨个遍历用户输入字符
    switch ( v6 )
    {
      case 'W':
        position += 10;                     // W向下走
        break;
      case 'S':
        position -= 10;                     // S向上走
        break;
      case 'A':
        ++position;                         // A向右走
        break;
      case 'D':
        --position;                         // D向左走
        break;
      case 'Q':
        if ( !has_num )     判断是否为0，如果为0则退出，仅仅有两个Q
        {
          puts("\nNo,You're out of 'Q'.");
          return 0LL;
        }
        mark = position;                    // Q给mark变量赋值，
        map[position] = 2;                  // 同时更改当前位置值为2
        --has_num;    数量减一，全局变量，初赋值为2
        break;
    }
    if ( v7 == 'Q' && (unsigned int)function() || map[position] > 10 )// 失败条件
                                            // v7也就是上上次按键为Q 并且 function函数返回真
                                            // 或者
                                            // 地图map的当前位置数值大于10
    {
      failed(v4);                           // 失败
      return 0LL;
    }
    if ( map[position] == 3 )               // 可得知终点是3
    {
      success();                            // 成功
      return 0LL;
    }
  }
  result = 0LL;
}
else
{
  puts("I don't think you got the right route....");
  result = 0LL;
}
return result;
}
```

很典型的一个迷宫题目，终点是3，墙是>10的随机数。使用position定位，是全局变量会初始化为0

根据上下走移动10可以得到迷宫的宽度为10

```
data:0000000000202020 ; _BYTE map[112]
data:0000000000202020 map             db 1, 40h, 3Ch, 31h, 1, 19h, 57h, 58h, 36h, 48h, 2 dup(1)
data:0000000000202020                                          ; DATA XREF: main+169↑o
data:0000000000202020                                          ; main+193↑o ...
data:0000000000202020                 db 21h, 38h, 2 dup(1), 4Ah, 4Eh, 35h, 28h, 48h, 1, 4Ch
data:0000000000202020                 db 23h, 18h, 1, 3Eh, 39h, 17h, 42h, 10h, 1, 24h, 2Fh, 5Ah
data:0000000000202020                 db 1, 30h, 3Ch, 20h, 57h, 2Ch, 1, 53h, 27h, 21h, 4 dup(1)
data:0000000000202020                 db 38h, 51h, 1, 2Ch, 1Bh, 1Eh, 35h, 4Bh, 32h, 1, 23h, 42h
data:0000000000202020                 db 5 dup(1), 46h, 57h, 1, 0Ch, 17h, 25h, 2Ah, 10h, 1Eh
data:0000000000202020                 db 1, 40h, 17h, 40h, 2Ah, 0Eh, 16h, 24h, 4Bh, 0Eh, 3Dh
data:0000000000202020                 db 28h, 24h, 1, 47h, 28h, 37h, 2Fh, 0Bh, 0Ch, 37h, 3Fh
data:0000000000202020                 db 31h, 1, 3, 0Ch dup(0)
data:0000000000202090 ; _DWORD arr[912]
```

112的数组，结尾有0Ch个0，也就是长度100

10*10的迷宫

比寻常的迷宫多了一个Q键，分析一下function函数

```
IDA View-A   Pseudocode-A   Hex View-1   Structures   Enums   Impo
signed __int64 sub_BB6()
{
  int v0; // ST04_4
  signed int i; // [rsp+0h] [rbp-8h]

  for ( i = 0; i <= 4; ++i )
  {
    v0 = arr[i] + mark;                    // v0等于 输入Q后标记上mark的格子 加上数组arr的前五位
    map[v0] = 1;                           // 将其赋值为1
    if ( v0 == position )                  // 如果v0恰好等于角色所在位置  返回1
                                           // 即进入失败函数
      return 1LL;
  }
  return 0LL;
}
```

看一下arr数组的前五位

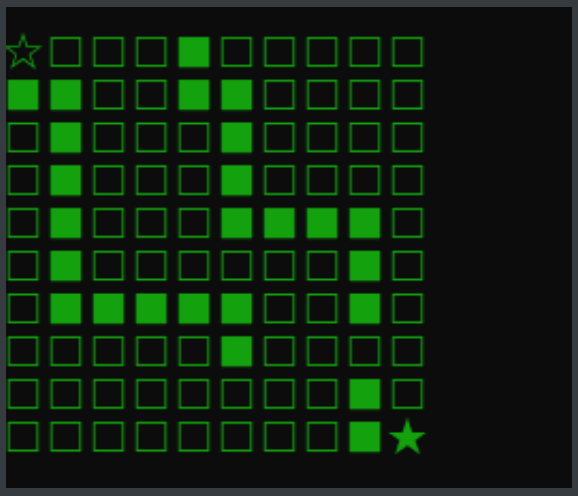
```
.data:0000000000202090 ; _DWORD arr[912]
.data:0000000000202090 arr            dd 0FFFFFFF6h, 0FFFFFFFFh, 0, 1, 0Ah, 7 dup(0), 4
.data:0000000000202090                                          ; DATA XREF: function+21
```

arr前五位为-10,-1,0,1,10 也就是mark位置和其位置的上下左右

会将这五个格子修改成1也就是路，同时检测到人物在这五个位置 就失败

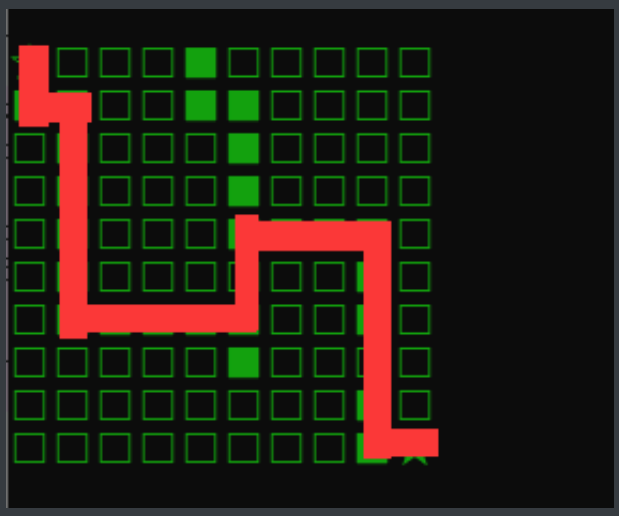
进入function函数的条件是v7为Q也就是上上步为Q，输入Q后有两步的移动时间。

先提取一下map得到迷宫，将数值1替换为实心方块，初始位置为0替换为空心星星，终点数值3替换为实心星星，大于10的墙替换为空心方块



得到的地图没有路可以从起点到终点，这时候就要用Q键来开辟道路了，

Q只能用两次 选好路线



但是答案不对，动调查看map变了，发现init_array执行了一个sub_8AA函数

```
init_array:0000000000201D80 _init_array     segment para public 'DATA' use64
init_array:0000000000201D80                 assume cs:_init_array
init_array:0000000000201D80                 ;org 201D80h
init_array:0000000000201D80 off_201D80      dq offset sub_8A0       ; DATA XREF: LOAD:00000000000000F8↑o
init_array:0000000000201D80                                         ; LOAD:0000000000000210↑o ...
init_array:0000000000201D88                 dq offset sub_8AA
init_array:0000000000201D88 _init_array     ends
init_array:0000000000201D88
ini_array:0000000000201D90 ; ELF Termination Function Table
ini_array:0000000000201D90 ; ===========================================================================
```

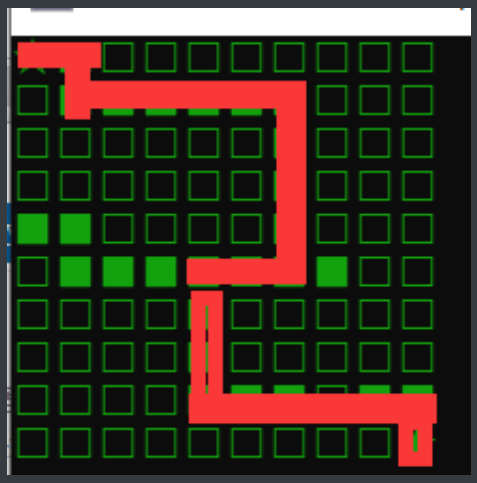
这个函数对map进行了更改

```
Instruction  Data  Unexplored  External symbol
IDA View-A   Pseudocode-C   Pseudocode-B   P
```

```c
char *sub_8AA()
{
  char *result; // rax
  char v1; // ST0C_1
  signed int i; // [rsp+0h] [rbp-14h]
  signed int j; // [rsp+4h] [rbp-10h]
  char *v4; // [rsp+Ch] [rbp-8h]

  result = &(*off_202ED0)[99];
  v4 = &(*off_202ED0)[99];
  for ( i = 0; i <= 9; ++i )
  {
    result = (char *)(unsigned int)i;
    for ( j = i; j <= 9; ++j )
    {
      v1 = v4[-10 * i - j];
      v4[-10 * i - j] = v4[-10 * j - i];
      result = &v4[-10 * j - i];
      *result = v1;
    }
  }
  return result;
}
```

程序动调再重新提取一下map数组



使用Q后要走两步躲开，小心角色死亡

也就是没有路放置Q后要后退两步躲开范围

```
flag{AWAAAAAWWWWQSSWWDDWWWAAQDDAAAAAW}
```