



Midterm Project: Nethack

实验报告

17343107 王明业

一、 代码运行

1. 集成开发工具 (IDE) : Xcode Version 9.2 (9C40b)
2. 头文件 :

```
#include <iostream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include <termios.h>
```

```
#include <iostream>
```

```
#include <string>
```

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

```
#include <time.h>
```

```
#include <termios.h>
```



特别说明 :

其中 `termios.h` 为 Linux 环境下串口驱动头文件, 用于实现 `_getch()` 函数, 即一个不回显函数, 当用户按下某个字符时, 函数自动读取, 无需按回车。

一般来说, `_getch()` 函数在 `conio.h` 头文件中。但 `conio.h` 不是 C 标准库中的头文件, 在 C standard library, ISO C 和 POSIX 标准中均没有定义。大部分 DOS, Windows 3.x, Phar Lap, DOSX, OS/2 or Win32 平台上的 C 编译器提供 `conio.h` 文件, UNIX 和 Linux 平台的 c 编译器通常不包含此头文件。

向用 Xcode 创建的 project 中自行添加 `conio.h` 文件, 运行时提示缺少 `_mingw.h` 文件, 继续自行添加 `_mingw.h` 文件后, 仍报错 :



 _mingw.h	A
 conio.h	A

Undefined symbols for architecture x86_64:
"__getch", referenced from:
Game::movearound() in main.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)

为在 Xcode 中使用 `_getch()` 函数，使用 `termios.h` 头文件，并在代码中利用 `termios.h` 自行定义 `_getch()` 函数：

```
char _getch()
{
    struct termios stored_settings;
    struct termios new_settings;
    tcgetattr(0, &stored_settings);
    new_settings = stored_settings;
    new_settings.c_lflag &= (~ICANON);
    new_settings.c_cc[VTIME] = 0;
    new_settings.c_cc[VMIN] = 1;
    tcsetattr(0, TCSANOW, &new_settings);

    char c;

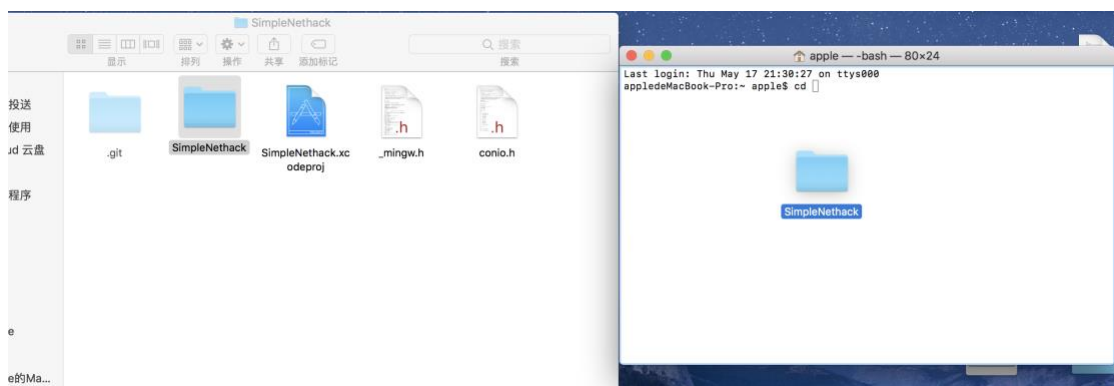
    c = getchar();

    tcsetattr(0, TCSANOW, &stored_settings);

    return c;
}
```

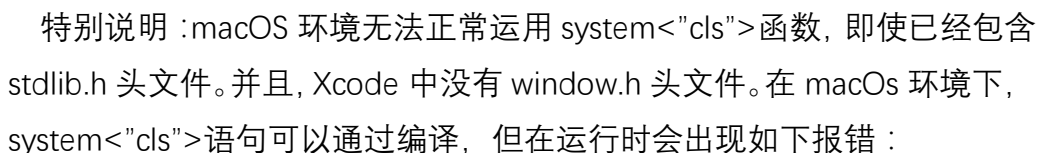
3. g++ 指令编译与运行：

I. 打开 macOS 系统自带的终端（Terminal），输入“`cd`”，然后输入源代码所在目录（可通过将源代码所在文件夹用鼠标拖入终端窗口实现）。



II. 然后输入“`g++`”后跟所有源代码中 `.cpp` 文件的文件名，可用空格分隔，完成后回车。此后，源代码所在文件夹会出现 `a.out` 文件。其中的 `.o` 文件，即对象文件（object file），内含汇编码，是由源码向机器码的过渡。

```
[appleMacBook-Pro:~ apple$ cd /Applications/SimpleNethack/SimpleNethack g++ main.cpp Player.cpp Equipment.cpp ]
Map.cpp Apple.cpp Monster.cpp
```



而用 Xcode 或终端运行程序时，由于极快的换行速度，该语句的出现并未对游戏过程造成影响。



二、游戏玩法

地图总大小为 30 行 50 列。四个房间，每个房间 10 行 20 列，相邻房间互通。
'-'代表墙，'#'代表门与路。

```
Player is @
Current ATK: 1
Current DEF: 0
Current HP : 3
Equipment  : Sword*0 Shield*0
Location   : (1,1)
Monster ATK: 2
Monster HP : 1
Floor      : 1
```

A 4x4 grid of 16 squares. The symbols and patterns are as follows:

@			W
		#####	
	&		
#	#	#	#
#	#	#	#
#	#	#	#
#	#	#	#
#	#	#	#
#	#	#	#
#	#	#	#
#	#	#	#
#	#	#	#
		#####	
			!
			O

1.角色-玩家 Player：@

初始生命：3 初始攻击力：1 初始防御力：0

每轮初始位置：(1, 1)，即地图左上角

八方向移动：

Q 键向左上移动, A 键向左移动, Z 键向左下移动, X 键向下移动,
C 键向右下移动, D 键向右移动, E 键向右上移动, W 键向上移动,
S 键休息 (不动)。



玩家遇到怪物'W'则自动攻击，怪物损失相当于玩家攻击力的生命值，玩家损失 怪物攻击力减去玩家防御力 的生命值，若怪物攻击力小于等于玩家防御力，则玩家生命值不变。攻击结束后，若玩家生命值小于等于零（即玩家阵亡）或怪物生命值大于零（即怪物未阵亡），则游戏结束。

玩家遇到剑'!'则自动拾取，剑从地图上消失，玩家攻击力加一。

玩家遇到盾'&'则自动拾取，盾从地图上消失，玩家防御力加一。

玩家遇到果实'O'则自动拾取，玩家根据当前层数增加一定生命值，并进入新的一层。

2.角色-怪物 Monster：W

初始生命：1 生命值每 5 层永久加 3

初始攻击力：2 攻击力每 5 层永久加 6

初始防御力：0

每轮初始位置：随机出现，但不会和玩家在同一个房间

八方向移动：

自动移动，追踪玩家。

当怪物与玩家不在同一房间时——

怪物会寻找自己所在房间最近的门，为了让自己尽快到达玩家所在房间。

当怪物与玩家在同一房间时——

怪物会自动按最小距离追踪玩家。

怪物遇到玩家'@'，会进行如 1.中所述玩家与怪物的攻击判定。

怪物遇到剑'!'，会自动把剑销毁，剑从地图上消失，怪物属性不变。

怪物遇到盾'&'，会自动把盾销毁，盾从地图上消失，怪物属性不变。

4. 道具-剑 Sword：!

每轮初始位置：随机出现

玩家拾取后攻击力加 1



5. 道具-盾 Shield : &

每轮初始位置：随机出现

玩家拾取后防御力加 1

6. 道具-果实 Apple : O













每轮初始位置：(48, 28) 即地图右下角

玩家拾取后，玩家生命值加 1。每第五层，果实所加生命值在该层增至 5；每第十层，果实所加生命值在该层增至 10。玩家拾取后，即视为通关本层，立即进入下一层。

三、 设计思路

1. 封装：

将要素分为 地图(Map)、玩家(Player)、怪物(Monster)、装备(Equipment)、果实(Apple)、玩法(Game)，分别用类封装以上 6 个要素。

 Monster.cpp	A
 Apple.cpp	A
 Equipment.cpp	A
 Player.cpp	A
 Map.cpp	A
 Apple.h	A
 Equipment.h	A
 Player.h	A
 Monster.h	A
 Game.h	A
 Map.h	A
 main.cpp	M

所有类各自的公有函数都可以实现读取、修改类成员变量。



2. 运算符重载的应用：

I. 玩家信息的输出：

```
ostream &operator<<( ostream &output,const Player &P )
{
    output<<"Player is "<<P.player<<endl;
    output<<"Current ATK: "<<P.atk<<endl;
    output<<"Current DEF: "<<P.def<<endl;
    output<<"Current HP : "<<P.hp<<endl;
    output<<"Equipment : "<<"Sword*<<P.equsword<<" Shield*<<P.equshield<<endl;
    output<<"Location : "<< "("<<P.row<<","<<P.column<<")"<<endl;
    return output;
}
```

II. 怪物信息的输出：

```
ostream &operator<<( ostream &output,const Monster &M )
{
    output<<"Monster ATK: "<<M.atk<<endl;
    if(M.column!=-1)
        output<<"Monster HP : "<<M.hp<<endl;
    else
        output<<"Monster HP : "<<"0"<<endl;
    return output;
}
```

四、 代码转换-Windows 环境下适用

1. 取消 `termios.h` 头文件的使用，改为使用 `conio.h` 头文件。
2. 利用 Visual Studio 的高级保存功能，将源代码文件文本转化以 Unicode 形式文本。

五、 其他

由于怪物、装备随机位置生成以及怪物强大的成长算法和八方向自动追踪算法，此游戏玩家必定会输，只能靠运气和摸索得到的技巧爬到高层。

还可考虑增加道具-金币，不定层、随机位置出现，并配套商店系统，可用金币购买装备。

也可增加地图的复杂性（或每层随机地图），增多装备、怪物的种类来提高可玩性。