

# 软件测试与分析-黑盒测试

软件测试 黑盒测试 脚本 shell

2018.11.28

张丽玮 2016K8009929029

## 测试要求

1. 输入"D M Y"返回当天是星期几（范围从1900年到2500年）
2. 对于输入的形式要有明确的提示
3. 当输入不合法的时候需要提醒用户
4. 程序不应该停止运行（卡住、关闭等）当输入不合法的时候
5. 黑盒方法设计测试用例

## 测试方法分析

1. 由于输入不合法时没有特定输出要求，因此暂时认为可以手动测试不合法输入，之后再脚本进行压力测试。
2. 由于1900-2500年总体数据量过于庞大，因此不选择全部天数测试的全部遍历测试方法，但是可以考虑以下方法
  - 回溯算法
  - 随机测试方法
  - 组合测试方法
3. 数据量较大，考虑用脚本实现。而关于如何生成测试数据，我应用了现成的excel软件功能，只需要更改单元格格式，就可以输出对应日期是星期几，并且这样生成csv文件对于之后脚本处理较为方便。

## 测试算法分析

1. 全部遍历算法：数据量过大，不进行选择

2. 回溯算法：

实际上考虑回溯算法，是因为日期和星期之间是有规律存在的，或者说是可以构成循环的。

一周七天，一年365或者366天，由于4年一个闰年，因此以年来计算，28年将会产生一个“星期-日期”的循环。也就是说，我的“D M Y”和“D M Y+28”对应星期几应该是相同的。既然如此，我们就可以采用回溯算法，给定一个日期，然后在年份上加减28的倍数，从而计算是否相等。

3. 随机测试方法：

非常显然，由于我之前分析过28年一个周期，所以这里可以只随机选择前28年中的任意一天，然后年份全部一直+28直到2500年。同样，这个也可以在excel中用随机生成日期的公式完成。

4. 组合测试方法：

我认为这个针对这个黑盒测试，组合测试没有必要，因为其实它本身就是一个组合测试——日、月、年的三元素组合。因此我觉得这样的输入数据难以分组，分组形式只可能是无效输入和有效输入组合，这个手动测试就能发现问题。

---

## 具体实现

---

### 无效输入测试

- 整型无效

```

Enter the DATE MONTH and YEAR: -1 -1 -1
Year out of range.
The year range is from 1900 to 2500.
Enter the DATE MONTH and YEAR: -1 0 0
Year out of range.
The year range is from 1900 to 2500.
Enter the DATE MONTH and YEAR: 32 13 2500
Month should be in range from 1 to 12.
Enter the DATE MONTH and YEAR: 32 1 2500
Invalid date in the 1th month.
The maximum date value is 31!
Enter the DATE MONTH and YEAR: 29 2 1901
Invalid date in the 2nd month of non leap year.
The maximum date value is 28!
Enter the DATE MONTH and YEAR:

```

可以报错从年月日依次报错，并且会提示某月最多只有多少天，月份是1-12，不会出现卡死、关闭等现象。

```

E:\school\computer\软件测试与分析\SoftwareTest\findday\td
Enter the DATE MONTH and YEAR: 0 0 1900
Month should be in range from 1 to 12.
Enter the DATE MONTH and YEAR: -1 11 1900

The Day is Tuesday.
Thanks for using my program.
Press any key to exit.

```

但是如果年份月份符合要求时，判断日期会出错。当日期 $\leq 0$ 时并不会报错，反而会输出结果。大体可以推断这是由于日期只进行了上界判断而没有判断下界。【bug1】

```

Enter the DATE MONTH and YEAR: 29 2 2100

The Day is Monday.
Thanks for using my program.
Press any key to exit.

```

闰年问题：普通闰年无误，但是世纪闰年有误。【bug2】

- 无输入  
无输入，一直空格回车时不会有任何输出（windows、linux都是）
- 输入类型无效



```
cod@cod-VirtualBox: /mnt/shared/shell$  
[10209]PASS!!!!!!  
[10210]PASS!!!!!!  
[10211]PASS!!!!!!  
[10212]PASS!!!!!!  
[10213]PASS!!!!!!  
[10214]PASS!!!!!!  
[10215]PASS!!!!!!  
[10216]PASS!!!!!!  
[10217]PASS!!!!!!  
[10218]PASS!!!!!!  
[10219]PASS!!!!!!  
[10220]PASS!!!!!!  
[10221]PASS!!!!!!  
[10222]PASS!!!!!!  
[10223]PASS!!!!!!  
[10224]PASS!!!!!!  
[10225]PASS!!!!!!  
[10226]PASS!!!!!!  
[10227]PASS!!!!!!  
[10228]PASS!!!!!!  
[10229]PASS!!!!!!  
[10230]PASS!!!!!!  
[10231]PASS!!!!!!  
cod@cod-VirtualBox: /mnt/shared/shell$
```

10231个测试用例全部pass，没有生成diffline.txt文件

- 随机测试

随机测试是在excel里面随机生成1000个1900-2500范围内的有效日期，然后用脚本进行测试。

· 第一组1000个测试数据：

```
[988]13 11 2437 reference:The Day is Friday result:The Day is Monday
[989]22 5 2382 reference:The Day is Saturday result:The Day is Tuesday
[990]PASS!!!!!!
[991]19 4 2138 reference:The Day is Saturday result:The Day is Sunday
[992]7 8 2236 reference:The Day is Sunday result:The Day is Tuesday
[993]11 8 2186 reference:The Day is Friday result:The Day is Saturday
[994]PASS!!!!!!
[995]25 4 2148 reference:The Day is Thursday result:The Day is Friday
[996]PASS!!!!!!
[997]23 11 2467 reference:The Day is Wednesday result:The Day is Saturday
[998]PASS!!!!!!
[999]9 12 2490 reference:The Day is Saturday result:The Day is Tuesday
[1000]29 3 2252 reference:The Day is Monday result:The Day is Wednesday
cod@cod-VirtualBox:/mnt/shared/shell$
```

查看diffline.txt的错误输出文本，共有678个错误，即这次的错误率是678/1000

接下来开始分析错误日期：

```
[976]7 2 2410 reference:The Day is Sunday result:The Day is Wednesday
[977]22 5 2328 reference:The Day is Tuesday result:The Day is Friday
[979]11 12 2223 reference:The Day is Thursday result:The Day is Saturday
[980]17 6 2295 reference:The Day is Monday result:The Day is Wednesday
[982]22 5 2489 reference:The Day is Sunday result:The Day is Wednesday
[984]19 1 2482 reference:The Day is Monday result:The Day is Thursday
[985]15 7 2264 reference:The Day is Friday result:The Day is Sunday
[987]25 10 2158 reference:The Day is Wednesday result:The Day is Thursday
[988]13 11 2437 reference:The Day is Friday result:The Day is Monday
[989]22 5 2382 reference:The Day is Saturday result:The Day is Tuesday
[991]19 4 2138 reference:The Day is Saturday result:The Day is Sunday
[992]7 8 2236 reference:The Day is Sunday result:The Day is Tuesday
[993]11 8 2186 reference:The Day is Friday result:The Day is Saturday
[995]25 4 2148 reference:The Day is Thursday result:The Day is Friday
[997]23 11 2467 reference:The Day is Wednesday result:The Day is Saturday
[999]9 12 2490 reference:The Day is Saturday result:The Day is Tuesday
[1000]29 3 2252 reference:The Day is Monday result:The Day is Wednesday
```

运用查询发现，所有错误日期都在2000年之后，不存在19xx的日期错误。那么初步猜测，1900-2000并没有错误。因此接下来进行分类测试：1900-2000年间和2000-2500年间

· 第二组1900-2000年间1000个测试数据：

```
cod@cod-VirtualBox: /mnt/shared/shell$
[978]PASS!!!!!!
[979]PASS!!!!!!
[980]PASS!!!!!!
[981]PASS!!!!!!
[982]PASS!!!!!!
[983]PASS!!!!!!
[984]PASS!!!!!!
[985]PASS!!!!!!
[986]PASS!!!!!!
[987]PASS!!!!!!
[988]PASS!!!!!!
[989]PASS!!!!!!
[990]PASS!!!!!!
[991]PASS!!!!!!
[992]PASS!!!!!!
[993]PASS!!!!!!
[994]PASS!!!!!!
[995]PASS!!!!!!
[996]PASS!!!!!!
[997]PASS!!!!!!
[998]PASS!!!!!!
[999]PASS!!!!!!
[1000]PASS!!!!!!
cod@cod-VirtualBox: /mnt/shared/shell$
```

1000个测试数据全部pass，也就是100%的通过率。再结合前28年全部pass的情况，现在有充足理由认为1900-2000年间全部正确没有bug。

· 第三组2000-2500年间1000个测试数据：

```
[990]15 11 2420 reference:The Day is Sunday result:The Day is Wednesday
[991]10 2 2240 reference:The Day is Monday result:The Day is Wednesday
[992]5 3 2412 reference:The Day is Monday result:The Day is Thursday
[993]20 10 2301 reference:The Day is Sunday result:The Day is Wednesday
[994]PASS!!!!!!
[995]PASS!!!!!!
[996]PASS!!!!!!
[997]31 3 2404 reference:The Day is Wednesday result:The Day is Saturday
[998]31 5 2391 reference:The Day is Friday result:The Day is Monday
[999]4 10 2226 reference:The Day is Wednesday result:The Day is Friday
[1000]15 12 2377 reference:The Day is Thursday result:The Day is Sunday
cod@cod-VirtualBox: /mnt/shared/shell$
```

可以发现这一次有相当多的错误情况。而查看diffline.txt的错误报告，共有809个错误，错误率是809/1000

继续分析错误日期：



```

796 [983]22 8 2253 reference:The Day is Monday result:The Day is Wednesday
797 [984]17 3 2204 reference:The Day is Saturday result:The Day is Monday
798 [985]9 10 2197 reference:The Day is Monday result:The Day is Tuesday
799 [986]6 2 2182 reference:The Day is Wednesday result:The Day is Thursday
800 [987]28 1 2326 reference:The Day is Thursday result:The Day is Sunday
801 [988]7 1 2146 reference:The Day is Friday result:The Day is Saturday
802 [990]15 11 2420 reference:The Day is Sunday result:The Day is Wednesday
803 [991]10 2 2240 reference:The Day is Monday result:The Day is Wednesday
804 [992]5 3 2412 reference:The Day is Monday result:The Day is Thursday
805 [993]20 10 2301 reference:The Day is Sunday result:The Day is Wednesday
806 [997]31 3 2404 reference:The Day is Wednesday result:The Day is Saturday
807 [998]31 5 2391 reference:The Day is Friday result:The Day is Monday
808 [999]4 10 2226 reference:The Day is Wednesday result:The Day is Friday
809 [1000]15 12 2377 reference:The Day is Thursday result:The Day is Sunday
810

```

发现并没有2000-2100年间的错误日期。按照同样的方法再次缩小范围。

· 第四组2000-2100年间1000个测试数据:

全部pass，认为这之间没有bug。

· 第五组2100-2500年间1000个测试数据：

```

[991]15 11 2388 reference:The Day is Tuesday result:The Day is Friday
[992]11 12 2334 reference:The Day is Tuesday result:The Day is Friday
[993]7 5 2427 reference:The Day is Friday result:The Day is Monday
[994]30 4 2309 reference:The Day is Friday result:The Day is Monday
[995]25 10 2281 reference:The Day is Tuesday result:The Day is Thursday
[996]5 7 2293 reference:The Day is Wednesday result:The Day is Friday
[997]15 6 2170 reference:The Day is Friday result:The Day is Saturday
[998]10 6 2457 reference:The Day is Sunday result:The Day is Wednesday
[999]25 2 2318 reference:The Day is Monday result:The Day is Thursday
[1000]3 8 2496 reference:The Day is Friday result:The Day is Monday
cod@cod-VirtualBox:/mnt/shared/shell$

```

明显错误更多了，这时候再查看diffline.txt，共有1000个错误，错误率1000/1000



```
[985]21 5 2498 reference:The Day is Wednesday result:The Day is Saturday
[986]9 11 2129 reference:The Day is Wednesday result:The Day is Thursday
[987]7 12 2293 reference:The Day is Thursday result:The Day is Saturday
[988]21 7 2375 reference:The Day is Monday result:The Day is Thursday
[989]7 8 2103 reference:The Day is Tuesday result:The Day is Wednesday
[990]11 1 2176 reference:The Day is Thursday result:The Day is Friday
[991]15 11 2388 reference:The Day is Tuesday result:The Day is Friday
[992]11 12 2334 reference:The Day is Tuesday result:The Day is Friday
[993]7 5 2427 reference:The Day is Friday result:The Day is Monday
[994]30 4 2309 reference:The Day is Friday result:The Day is Monday
[995]25 10 2281 reference:The Day is Tuesday result:The Day is Thursday
[996]5 7 2293 reference:The Day is Wednesday result:The Day is Friday
[997]15 6 2170 reference:The Day is Friday result:The Day is Saturday
[998]10 6 2457 reference:The Day is Sunday result:The Day is Wednesday
[999]25 2 2318 reference:The Day is Monday result:The Day is Thursday
[1000]3 8 2496 reference:The Day is Friday result:The Day is Monday
```

百分之一百的错误率，对于随机测试来说可以基本认为这表示从2100到2500年间的所有日期都是错的。

那么为什么会发生这种错误？根据之前世纪闰年有误，推测是由世纪闰年导致，因此接着选取几个年份分析。

· 第六组2300年的全部365个测试数据

```
result:The Day is Tuesday
[57]26 2 2300 reference:The Day is Monday
result:The Day is Wednesday
[58]27 2 2300 reference:The Day is Tuesday
result:The Day is Thursday
[59]28 2 2300 reference:The Day is Wednesday
result:The Day is Friday
[60]1 3 2300 reference:The Day is Thursday
result:The Day is Sunday
[61]2 3 2300 reference:The Day is Friday
result:The Day is Monday
[62]3 3 2300 reference:The Day is Saturday
result:The Day is Tuesday
```

错误非常有规律，可以看到在2月28日前后，从全部向后推移两天变成全部向后推移三天。因此可以判断这个是由于findday将2300年当做闰年，增加计算了2月29日的日期，导致的星期整体平移的结果。根据这个猜测再测另外几个世纪闰年。

· 第七组2100年的全部365个测试数据

```
[60]1 3 2100 reference:The Day is Monday
result:The Day is Tuesday
[61]2 3 2100 reference:The Day is Tuesday
result:The Day is Wednesday
[62]3 3 2100 reference:The Day is Wednesday
result:The Day is Thursday
[63]4 3 2100 reference:The Day is Thursday
result:The Day is Friday
[64]5 3 2100 reference:The Day is Friday
result:The Day is Saturday
[65]6 3 2100 reference:The Day is Saturday
result:The Day is Sunday
[66]7 3 2100 reference:The Day is Sunday
```

同样非常有规律，实际上是从3月1日开始出错，并且所有日期都向后移一天，因此可以判定是世纪闰年2100年的2月29日多出来了。

· 第八组2200年的全部365个测试数据

```
113 [57]26 2 2200 reference:The Day is Wednesday
114 result:The Day is Thursday
115 [58]27 2 2200 reference:The Day is Thursday
116 result:The Day is Friday
117 [59]28 2 2200 reference:The Day is Friday
118 result:The Day is Saturday
119 [60]1 3 2200 reference:The Day is Saturday
120 result:The Day is Monday
121 [61]2 3 2200 reference:The Day is Sunday
122 result:The Day is Tuesday
```

同样非常有规律，可以看到2月28日前后，从全部向后推移一天变成全部向后推移两天。

· 第九组2400年的全部365个测试数据

```

113 [57]26 2 2400 reference:The Day is Saturday
114 result:The Day is Tuesday
115 [58]27 2 2400 reference:The Day is Sunday
116 result:The Day is Wednesday
117 [59]28 2 2400 reference:The Day is Monday
118 result:The Day is Thursday
119 [60]29 2 2400 reference:The Day is Tuesday
120 result:The Day is Friday
121 [61]1 3 2400 reference:The Day is Wednesday
122 result:The Day is Saturday
123 [62]2 3 2400 reference:The Day is Thursday
124 result:The Day is Sunday
125 [63]3 3 2400 reference:The Day is Friday

```

由于2400年是闰年，所以所有日期仍然是向后推移3天

· 第十组2500年的全部365个测试数据

```

114 result:The Day is Monday
115 [58]27 2 2500 reference:The Day is Saturday
116 result:The Day is Tuesday
117 [59]28 2 2500 reference:The Day is Sunday
118 result:The Day is Wednesday
119 [60]1 3 2500 reference:The Day is Monday
120 result:The Day is Friday
121 [61]2 3 2500 reference:The Day is Tuesday
122 result:The Day is Saturday
123 [62]3 3 2500 reference:The Day is Wednesday
124 result:The Day is Sunday

```

2500年的2月29日同样多出，所以从前后从向后推移三天变为推移四天

### 【bug】

· 现在初步得到猜测：1900-2100年间的日期对应星期都是正确的，但是2100-2500年间有错误，并且错误有规律。是由2月29日世纪闰年导致，因此

2100年3月1日-2200年2月28日 全部向后推移一天

2200年3月1日-2300年2月28日 全部向后推移两天

2300年3月1日-2500年2月28日 全部向后推移三天

2500年3月1日-2500年12月31日 全部向后推移四天

· 实际上在压力测试下可能出现了死循环卡死问题：

在进行random测试时脚本一切正常，但是进行单年测试时，出现了导出文件中输出语句死循环的情况

test\_temp

2018/11/29 4:26

文本文档

4,606,283...

输入无误，生成的test\_in和test\_out也没有问题。

暂时无法定位是怎样出的问题，猜测是在特定条件下的压力测试可能出现死循环情况。

## 测试脚本

实验过程中均采用我自己用shell编写的简易脚本，在linux上运行

· 一键完成脚本test0

```
1.
2.  #!/bin/sh
3.  cat $1.csv | awk -F, '{ print $1; }' > test_in.txt
4.  cat $1.csv | awk -F, '{ print $2; }' > test_out.txt
5.
6.      if [ -f "test_result.txt" ];
7.      then
8.          rm test_result.txt
9.      fi
10.  ##生成对应的测试文件输入和输出
11.  #i=0;
12.  cat test_in.txt | while read str_in #逐行获取输入
13.  do
14.      output=`./test.out $str_in >> test_temp.txt 2>&1`
15.  done
16.  cat test_temp.txt | grep "The Day" | while read str
17.  do
18.      echo ${str%%.} >> test_result.txt 2>&1
19.  done
20.
21.
22.
23.      if [ -f "test_temp.txt" ];
24.      then
25.          rm test_temp.txt
26.      fi
27.      if [ -f "diffline.txt" ];
28.      then
```

```

29.         rm diffline.txt
30.     fi
31.     if [ -f "sameline.txt" ];
32.     then
33.         rm sameline.txt
34.     fi
35.
36.
37.     i=0
38.     cat test_result.txt | while read result
39.     do
40.         ((i=i+1))
41.         sed -n ${i}p test_out.txt | while read str_out
42.         do
43.             reference="The Day is $str_out"
44.             sed -n ${i}p test_in.txt | while read str_in
45.             do
46.                 #echo $output
47.                 if [ "$reference" \= "$result" ]
48.                 then
49.                     echo "[${i}]PASS!!!!!!"
50.                     echo "[${i}]$str_in reference:$reference" >> sameline.txt
51.                 else
52.                     echo "[${i}]$str_in reference:$reference result:$result" >> diffline.txt
53.                     echo "[${i}]$str_in reference:$reference result:$result"
54.                 fi
55.             done
56.         done
57.     done

```

使用方法是输入 ./test0.sh 测试数据csv表名

注：不要写csv后缀

生成的diffline.txt会标注所有错误的行数、日期、以及正确输出和错误输出

生成的sameline.txt则是所有正确的行数、日期、正确输出

· 分步式防卡死脚本test+test2

```

1.
2.     #! /bin/sh

```

```

3. cat $1.csv | awk -F, '{ print $1; }' > test_in.txt
4. cat $1.csv | awk -F, '{ print $2; }' > test_out.txt
5. ##生成对应的测试文件输入和输出
6. #i=0;
7. cat test_in.txt | while read str_in #逐行获取输入
8. do
9.     #echo "$str_in"
10.    output=`./test.out $str_in`
11. done

```

```

1.
2. #!/bin/sh
3. cat terminal.txt | grep "The Day" | while read str
4. do
5.     echo ${str%.*} >> test_result.txt
6. done
7.     if [ -f "test_temp.txt" ];
8.     then
9.         rm test_temp.txt
10.    fi
11.    if [ -f "diffline.txt" ];
12.    then
13.        rm diffline.txt
14.    fi
15.    if [ -f "sameline.txt" ];
16.    then
17.        rm sameline.txt
18.    fi
19.
20.
21.    i=0
22.    cat test_result.txt | while read result
23.    do
24.        ((i=i+1))
25.        sed -n ${i}p test_out.txt | while read str_out
26.        do
27.            reference="The Day is $str_out"
28.            sed -n ${i}p test_in.txt | while read str_in
29.            do
30.                #echo $output
31.                if [ "$reference" \= "$result" ]
32.                then
33.                    echo "[$i]PASS!!!!!!"
34.                else

```



```
35.         echo "[${i}]$str_in reference:$reference result:$result" >> diff1
        ine.txt
36.         echo "[${i}]$str_in reference:$reference result:$result"
37.     fi
38. done
39. done
40. done
```

使用方法：

script terminal.txt先打开终端日志记录

./test.sh 测试数据表名

exit退出终端日志

然后执行./test2.sh

最后生成的文件结果和test0一样。

这两个脚本只是防止一键完成时中途卡死，通过终端日志中转完成测试。

---

## 结果总结

### 发现的bug

1. 非整型类型的无效输入会导致死循环
2. 日期 $\leq 0$  但是月年正确时不会报错
3. 世纪闰年有误
4. 从2100-2500年间规律性出错
  - 2100年3月1日-2200年2月28日 全部向后推移一天
  - 2200年3月1日-2300年2月28日 全部向后推移两天
  - 2300年3月1日-2500年2月28日 全部向后推移三天
  - 2500年3月1日-2500年12月31日 全部向后推移四天
5. 不断进行同一年连续测试时可能会导致死循环

### 猜测bug原因

1. 没有进行类型判断，并没有处理整型以外的类型，导致读入错误进入死循环

2. 没有进行日期的下界判断，导致没有报错正常输出
3. 只进行了mod 4的判断但没有进行mod 10和mod 1000的判断，一律视为闰年
4. 由于世纪闰年错误导致的整体平移，并且2100-2500年间错误率为100%，整百年2月到3月的错误平移明显和2月29日有关。
5. 排查发现应该是csv文件中最后隐藏空行带了某种符号，而这种符号不符合findday的输入规范，因此和非整型输入一样开始死循环。最后处理方法是生成数据时多拉一行，再删除（下方单元格上移），就不再会死循环。

## 不足之处

1. 初始的28年计算显得多此一举。实际上这个实验，回溯算法并没有很好的效果
2. 随机算法随机性比较大，还是不能完全排除遗漏的错误可能（比如再1900-2100年间）
3. 最后单独分析年份的时候是取的百年，因为猜测是百年计更改的输出，但是不排除可能这中间的年份并不是这个规律。
4. 测试工程量还是比较大，实际上我是在手动进行数据的筛选工作，而更理想的情况是我们把这个也编入脚本，由脚本来进行筛选，这样效率更高，准确率也更高。
5. 实际上查资料发现一般万年历是从1900年1月1日为周一开始计算的，但是excel实际上是周日，而且1900年被认为是闰年，巧合的是这个软件也这么认为。所以说我这一段是全pass的，但如果是从按照一般万年历，应该从1900年1月1日到1900年2月28日的结果都是有误的，直到3月1日恢复正常。
6. 脚本是在linux下运行的，但是手动是在windows下进行的，实际上可能这两个平台上会有差异。
7. 脚本本身也可能存在错误。一开始判断比较结果是用了<,为了处理末尾的“.”,但是这样的结果会变成50%的错误率，非常匪夷所思。改成=判断之后，就一切正常了。

---

## 补充说明

除了文档外所有的包括脚本、数据、错误输出等文件全部打包  
文件架构如下：

- ```
1. SoftwareTest_findday
2. |—reveal.js
```

```
3.  |─test_data
4.  |   |─test_2100
5.  |   |─test_2202
6.  |   |─test_2300
7.  |   |─test_2400
8.  |   |─test_2500
9.  |   |─test_all
10. |   |─test_random1
11. |   |─test_random2
12. |   |─test_random3
13. |   |─test_random4
14. |   |─test_random5
15. |─test_result
16. |   |─diffline_2100
17. |   |─diffline_2200
18. |   |─diffline_2300
19. |   |─diffline_2400
20. |   |─diffline_2500
21. |   |─diffline_random1
22. |   |─diffline_random3
23. |   |─diffline_random5
24. |─test.exe
25. |─test.out
26. |─test0.sh
27. |─test.sh
28. |─test2.sh
29. |─test_in.txt
30. |─test_out.txt
31. |─test_result.txt
32. |─my_presentation.md
33. |─my_presentation.html
34. |─软件测试分析-黑盒测试.pdf
35. |─README.md
```

使用说明：由于测试的时候并没有分文件夹全都在一个目录下，因此如果要测试数据需要在写表名时加上path。

my\_presentation和reveal.js必须在同一目录下才能正常播放