Project0 实验报告

钱庭涵 10152130122

Introduction

在 part1 中我实现了 s1.sh 和 s2.sh 的功能：
s1.sh（创建目录，创建 txt 文件，使用重定向将字符串输入到 txt 文件中，将 txt 文件复制到新创建的这个目录下）
s2.sh（创建 txt 文件，在该文件中输入内容，这些内容是某目录下下固定字符开头的文件，包含名称、所有者、权限三个域，并且根据名称按字母从小到大排序，最后设置权限使得其他人只能读该文件，不能写或执行）

在 part2 中我利用 gdb 调试，找出并更正了 set_operation.c 中的 4 处 bug。

在 bonus1 中，使用 awk 输出指定的某行。

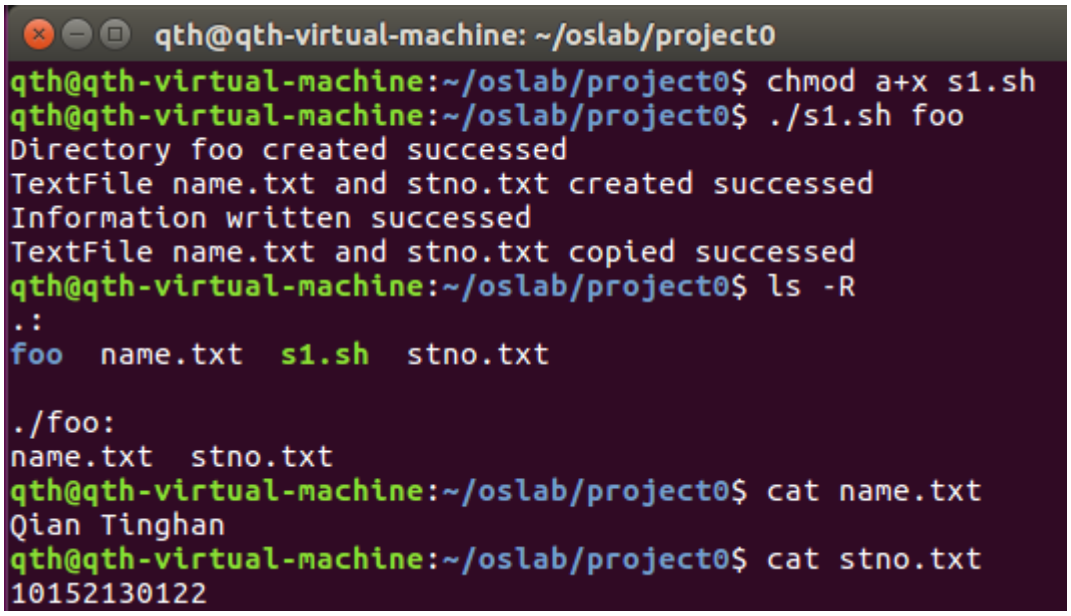
在 bonus2 中，使用 awk 实现列转行。

Part 1

s1.sh

```
#!/bin/sh

NewDirectoryName=$1 #将第一个变量赋值到NewDirectoryName里
mkdir $NewDirectoryName #创建一个新目录$NewDirectoryName
echo "Directory $NewDirectoryName created successed"
                    #输出字符串信息表示创建成功

touch name.txt #在当前目录下创建name.txt文件
touch stno.txt #在当前目录下创建stno.txt文件
echo "TextFile name.txt and stno.txt created successed" #创建成功

echo "Qian Tinghan" > name.txt #输入字符串信息到name.txt里
echo "10152130122" > stno.txt #输入字符串信息到stno.txt里
echo "Information written successed" #输入成功

cp name.txt $NewDirectoryName #将name.txt复制到$NewDirectoryName目录下
cp stno.txt $NewDirectoryName #将stno.txt复制到$NewDirectoryName目录下
echo "TextFile name.txt and stno.txt copied successed" #复制成功
```

chmod a+x s1.sh ： 给 s1.sh 文件添加权限，使其可以直接运行。
./s1.sh foo ： 运行 s1.sh 文件，同时输入变量值 foo。
然后用 ls -R 和 cat 命令检查结果。



s2.sh

touch output.txt ： 创建 output.txt 文件。

| 、xargs ： 将上一个命令的输出作为下一个命令的输入。
find /bin -name 'b*' ： 从/bin 目录下按照名称找出所有名称开头为 b 的文件。
grep '/bin/b*' ： 因为 bin 目录本身也是 b 开头的文件，所以要从剩下的文件的长格式字符串中找出所有含'/bin/b'的文件。

经过两次查找，现在剩下来的文件都是 bin 目录下名称为 b 开头的文件。

ls -l ： 显示剩下来的文件的长格式。
awk '{printf "%s %s %s\n", substr($9,6), $3, $1}' ： 按照格式输出，先输出第 9 个域第 6 个字符开始的字符串(即去掉路径前面的/bin/后的文件名称)，再输出第 3 个域（即 owner），再输出第 1 个域（即权限）。
sort -k1,1 -o output.txt ： 对剩下的文件排序，从第一个域开始开始，从第一个域结束（即仅对第一个域进行排序），默认按照字母从小到大排序，再将结果输出到 output.txt 中。

chmod 744 output.txt ： 赋予 output.txt 权限，owner 可以读、写、执行，group 和 others 只能读。

```
#!/bin/sh

touch output.txt
echo "TextFile output.txt created successed"

find /bin -name 'b*' | grep '/bin/b*' | xargs ls -l | awk '{printf "%s %s %s\n", substr($9,6), $3, $1}' | sort -k1,1 -o output.txt
echo "Information written successed"

chmod 744 output.txt
echo "Authority set successed"

```

Part 2

set_operation.c

首先编译，gcc -o set_operation set_operation.c 是对 set_operation.c 这个文件进行编译，并将生成的可执行文件命名为 set_operation。
按下回车键后，发现命令行报错，指出代码 26 行有错，正确的格式应该是 p->next 而不是 p->->next。

```
qth@qth-virtual-machine:~/oslab/project0$ gcc -o set_operation set_operation.c
set_operation.c: In function ‘check’:
set_operation.c:26:12: error: expected identifier before ‘->’ token
     if((p->->next)->number==num)
            ^
```

```
25      while(p!=NULL){
26        if((p->->next)->number==num)
27          sign=1;
```

将第一个错误改正后，再次编译，发现编译通过，然后输入./set_operation，运行可执行文件，输入集合 A 和 B 的各元素之后，发现报错“段错误”。

```
qth@qth-virtual-machine:~/oslab/project0$ gcc -o set_operation set_operation.c
qth@qth-virtual-machine:~/oslab/project0$ ./set_operation
--------------------------------
----Computing (A-B)union(B-A)----
--------------------------------
----input the number of elements of A: 3
1-th element: 3
2-th element: 4
3-th element: 5
----input the number of elements of B: 2
1-th element: 1
2-th element: 4
段错误 (核心已转储)
```

段错误可能是越界造成的，因为输入 A 和 B 的元素时并没有报错，可知 input A 和 input B 的过程中没有错误，下一个过程是 copy A to A2。

重新编译，加上-g 可以产生供 gdb 调试用的可执行文件，然后输入 gdb set_operation 命令，进入 gdb 模式。

```
qth@qth-virtual-machine:~/oslab/project0$ gcc -g -o set_operation set_operation.
c
qth@qth-virtual-machine:~/oslab/project0$ gdb set_operation
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from set_operation...done.
(gdb)
```

考虑到 copy A to A2 的过程中含有循环，因此将断点设置在第 90 行，输入命令 b 90。然后输入 r 运行，输入集合 A 和 B 的各元素。

```
(gdb) b 90
Breakpoint 1 at 0x400981: file set_operation.c, line 90.
(gdb) r
Starting program: /home/qth/oslab/project0/set_operation
--------------------------------
----Computing (A-B)union(B-A)----
--------------------------------
----input the number of elements of A: 3
1-th element: 3
2-th element: 4
3-th element: 5
----input the number of elements of B: 2
1-th element: 1
2-th element: 4

Breakpoint 1, main () at set_operation.c:90
90          p2=(struct node*)malloc(LEN);
```

观察 p3->number 这个变量，输入命令 p p3->number，得到该变量当前值 3，然后输入 c 进行一次循环，然后再输入命令得到 p3->number 的值为 4，重复上述过程，发现在第 4 次循环时，p3->number 的值已经找不到了，但是循环仍在进行，由此可知，这个循环中存在越界。

```
(gdb) p p3->number
$1 = 3
(gdb) c
Continuing.

Breakpoint 1, main () at set_operation.c:90
90          p2=(struct node*)malloc(LEN);
(gdb) p p3->number
$2 = 4
(gdb) c
Continuing.

Breakpoint 1, main () at set_operation.c:90
90          p2=(struct node*)malloc(LEN);
(gdb) p p3->number
$3 = 5
(gdb) c
Continuing.

Breakpoint 1, main () at set_operation.c:90
90          p2=(struct node*)malloc(LEN);
(gdb) p p3->number
Cannot access memory at address 0x0
(gdb)
```

观察第 89 行代码，发现 i 的取值范围应该是 0~A_size-1 才对，所以将‘=’去掉。

```
89      for(i=0;i<=A_size;i++){
90        p2=(struct node*)malloc(LEN);
91        p2->number=p3->number;
```

重新编译，再次运行，发现仍然报错“段错误”，进入 gdb 模式，先不设置断点，输入 r 运行，输入集合 A 和 B 的各元素，发现报错显示，第 26 行出现了段错误。

```
(gdb) r
Starting program: /home/qth/oslab/project0/set_operation
--------------------------------
----Computing (A-B)union(B-A)----
--------------------------------
----input the number of elements of A: 3
1-th element: 3
2-th element: 4
3-th element: 5
----input the number of elements of B: 2
1-th element: 1
2-th element: 4

Program received signal SIGSEGV, Segmentation fault.
0x0000000000400778 in check (num=3, head=0x603890) at set_operation.c:26
26          if((p->next)->number==num)
(gdb)
```

观察第 26 行代码，发现 if 条件语句是从(p->next)->number 开始判断的，也就是说头结点的 head->number 被落掉了，因此第 26 行应改为 if(p->number==num)。

```
24      p=head;
25      while(p!=NULL){
26        if((p->next)->number==num)
27          sign=1;
28        p=p->next;
29      }
```

重新编译，再次运行，发现得到的结果是{4,4}，并不是想要得到的最终结果，但是可以发现，这个结果正好与正确结果{3,5,1}相反，所以猜测是 compute A = A - B and B = B - A2 中的 check 函数返回值出错或判断出错。

```
qth@qth-virtual-machine:~/oslab/project0$ gcc -g -o set_operation set_operation.
c
qth@qth-virtual-machine:~/oslab/project0$ ./set_operation
--------------------------------
----Computing (A-B)union(B-A)----
--------------------------------
----input the number of elements of A: 3
1-th element: 3
2-th element: 4
3-th element: 5
----input the number of elements of B: 2
1-th element: 1
2-th element: 4
---- elements of (A-B)union(B-A) ----
1-th element: 4
2-th element: 4
sh: 1: pause: not found
```

将断点设置在第 107 行，输入命令 b 107。然后输入 r 运行，输入集合 A 和 B 的各元素。

```
(gdb) b 107
Breakpoint 1 at 0x400a09: file set_operation.c, line 107.
(gdb) r
Starting program: /home/qth/oslab/project0/set_operation
---------------------------------
----Computing (A-B)union(B-A)----
---------------------------------
----input the number of elements of A: 3
1-th element: 3
2-th element: 4
3-th element: 5
----input the number of elements of B: 2
1-th element: 1
2-th element: 4

Breakpoint 1, main () at set_operation.c:107
107         if(!check(p1->number,B_head)){ //if this element is in B
(gdb)
```

观察 check 函数的返回值，输入命令 p check(p1->number,B_head)，得到返回值为 0，因为集合 A 的第一个元素 3 确实不在 B 中，返回值 0 是正确结果，可知 check 函数本身没有问题。

```
Breakpoint 1, main () at set_operation.c:107
107         if(!check(p1->number,B_head)){ //if this element is in B
(gdb) p check(p1->number,B_head)
$1 = 0
```

输入命令 n，进行不进入函数的单步调试，发现程序进行到代码第 108 行，输入命令 p sign，得到此时 sign 的值为 0，输入 n 发现程序进入第 109 行。

```
(gdb) n
108             if(sign==0){
(gdb) p sign
$2 = 0
(gdb) n
109                 A_head=A_head->next;
(gdb)
```

观察代码发现，第 109 行接下来要执行的代码是将 A 的头结点去掉，但是 A 的第一个元素 3 并不在 B 中，正确的结果应该是保留 A 的头结点。此时 sign 的值为 0，所以正确执行时，下一步应该进入第 118 行和第 119 行。因此将第 107 行的取反符号‘!’去掉。

```
103     //////////////A'=A-B///////////////
104     p2=p1=A_head;
105     sign=0;
106     for(i=0;i<A_size;i++){ //p1!=NULL){
107       if(!check(p1->number,B_head)){ //if this element is in B
108         if(sign==0){
109           A_head=A_head->next;
110           p2=p1=A_head;
111         }
112         else{
113               p2->next=p1->next;
114               p1=p1->next;
115         }
116       }
117       else{
118             if(sign==0){
119               p1=p1->next;
120               sign=1;
121             }
122             else{
123               p2=p2->next;
124               p1=p1->next;
125             }
126       }
127     }
```

同理，在 compute B = B - A2 的过程中也出现了同样的错误。重新编译，进入 gdb 模式，设置断点在第 133 行。

```
(gdb) b 133
Breakpoint 1 at 0x400ac9: file set_operation.c, line 133.
(gdb) r
```

观察 check 函数的返回值，发现符合正确结果 0，再进行单步调试，发现进入了第 134 行和第 135 行，而不是应该进入的第 144 行和第 145 行，因此将第 133 行的取反符号‘!’去掉。

```
Breakpoint 1, main () at set_operation.c:133
133         if(!check(p1->number,A2_head)){
(gdb) p check(p1->number,A2_head)
$1 = 0
(gdb) n
134           if(sign==0){
(gdb) p sign
$2 = 0
(gdb) n
135             B_head=B_head->next;
(gdb)
```

```
132     for(i=0;i<B_size;i++){
133       if(!check(p1->number,A2_head)){
134         if(sign==0){
135           B_head=B_head->next;
136           p2=p1=B_head;
137         }
```

重新编译，再次运行，输入集合 A 和 B 的各元素，发现结果正确，代码改正完毕。

```
qth@qth-virtual-machine:~/oslab/project0$ gcc -g -o set_operation set_operation.
c
qth@qth-virtual-machine:~/oslab/project0$ ./set_operation
--------------------------------
----Computing (A-B)union(B-A)----
--------------------------------
----input the number of elements of A: 3
1-th element: 3
2-th element: 4
3-th element: 5
----input the number of elements of B: 2
1-th element: 1
2-th element: 4
---- elements of (A-B)union(B-A) ----
1-th element: 3
2-th element: 5
3-th element: 1
sh: 1: pause: not found
```

Bonus 1

bonus1.sh

运行结果：

```
qth@qth-virtual-machine:~/oslab/project0$ chmod a+x bonus1.sh
qth@qth-virtual-machine:~/oslab/project0$ ./bonus1.sh
Line 10
qth@qth-virtual-machine:~/oslab/project0$
```

Bonus 2

bonus2.sh

使用 awk，先通过循环将 bonus2.sh 中的字符串读入 awk 内置变量二维数组 a 中，再通过循环按先列后行输出。

运行结果：

```
qth@qth-virtual-machine:~/oslab/project0$ chmod a+x bonus2.sh
qth@qth-virtual-machine:~/oslab/project0$ ./bonus2.sh
name alice ryan
age 21 30
qth@qth-virtual-machine:~/oslab/project0$
```