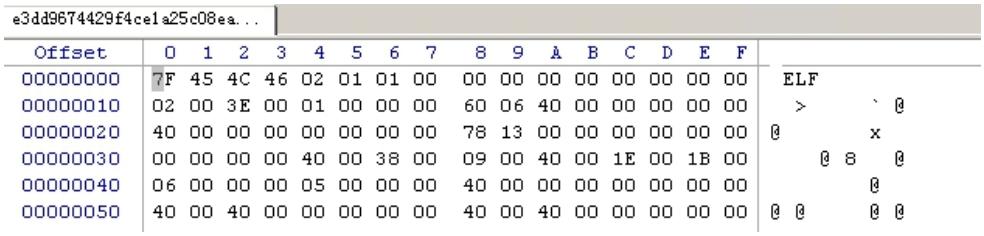
原题给了一个文件，丢进 WinHex 看文件头，分析文件类型，得到为 elf 文件。



直接丢进 IDA，查看文件反汇编是否正常，判断有无加壳，发现反汇编正常，无壳，打开字符串窗口查看字符串。

```
                .
; char s[]
s               db 'c61b68366edeb7bdce3c6820314b7498',0
                                        ; DATA XREF: main+25↑o
                                        ; main+3F↑r
                align 20h
                public t
; char t
t               db 53h                  ; DATA XREF: main+65↑w
                                        ; main+C9↑o ...
aHarifctf?????? db 'harifCTF{??????????????????????????????????}',0
                align 20h
                public u
u               db '********************************************',0
                                        ; DATA XREF: main+A5↑o
```

看到开头有个字符串 t 有 ctf 字样，猜测为 flag，由于 t 为字符串形式，db 53h 应该为转义错误，实际上 t 为 char(0x53)+harifCTF{??????????????????????????????????}，即 SharifCTF{??????????????????????????????????}。

接下来进入 main 函数，观察发现 main 函数实际可分三个部分：

1、开头部分，定义初值

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v3; // al@4
  int result; // eax@10
  __int64 v5; // rbx@10
  __int64 v6; // [sp+0h] [bp-40h]@0
  int i; // [sp+4h] [bp-3Ch]@7
  FILE *stream; // [sp+8h] [bp-38h]@7
  char filename[8]; // [sp+10h] [bp-30h]@7
  __int64 v10; // [sp+28h] [bp-18h]@1
```

2、结尾部分，将 flag 值写入文件 flag.txt 中

```
  strcpy(filename, "/tmp/flag.txt");
  stream = fopen(filename, "w");
  fprintf(stream, "%s\n", u, v6);
  for ( i = 0; i < strlen(&t); ++i )
  {
    fseek(stream, p[i], 0);
    fputc(*(&t + p[i]), stream);
    fseek(stream, 0LL, 0);
    fprintf(stream, "%s\n", u);
  }
  fclose(stream);
  remove(filename);
  result = 0;
  v5 = *MK_FP(__FS__, 40LL) ^ v10;
  return result;
}
```

3、中间部分，获得 flag 值

```
v10 = *MK_FP(__FS__, 40LL);
LODWORD(v6) = 0;
while ( (signed int)v6 < strlen(s) )
{
  if ( v6 & 1 )
    v3 = 1;
  else
    v3 = -1;
  *(&t + (signed int)v6 + 10) = s[(signed __int64)(signed int)v6] + v3;
  LODWORD(v6) = v6 + 1;
}
```

我们关心的是如何获得 flag 值，故只需要还原第三部分代码即可获得 flag 值，没必要还原写入文件的过程。

分析中间部分代码，发现其定义初值 v6，判断 v6 and 1 的结果，为真，则 v3=1，否则 v3=-1。之后将 t[v6+10]替换为 s[v6]+v3，此处需要注意 s 为字符串，而 v3 为整型，需要做数据类型转换才可以加和。（s 里面的(signed __int64)(signed int)就是原本此处做了数据类型转换。

利用 python 还原算法，需要注意的是，python 中字符串中的值不可被直接替换，需要将字符串转换为 list 后才可替换。完整代码如下：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#coding=utf-8
#上面几行不加会报错，，，

t="SharifCTF{??????????????????????????????????}"      #由文件本身获得的初值
s="c61b68366edeb7bdce3c6820314b7498"
v6=0
tlist=[]

for i in t:
    tlist.append(i)    #python中字符串无法替换单个字符，需要转换为list才可替换

while v6<len(s):      #原逻辑还原
        if(v6&1):
            v3=1
        else:
            v3=-1
        tlist[v6+10]=chr(ord(s[v6])+v3)
        v6=v6+1

flag="".join(tlist) #list转换回字符串，读着更舒服，

print(flag)
```

运行，获得 flag：SharifCTF{b70c59275fcfa8aebf2d5911223c6589}