

分析phase3的源码可知，其中定义了一个未初始化的全局数组PHASE3\_CODEBOOK，这显然是一个弱符号，因此我们可以在phase3\_patch.o中定义一个同名的符号并初始化，初始化之后这就是一个强符号，在链接时让链接器选择我们自己构造的强符号。因此剩下的问题就是，如何巧妙的构造这个数组，使其输出自己的学号。

我的学号是1190200910，因此要求构造的数组满足：

PHASE3\_CODEBOOK [(unsigned char)(cookie[0])] = ‘1’ = 31

PHASE3\_CODEBOOK [(unsigned char)(cookie[1])] = ‘1’ = 31

PHASE3\_CODEBOOK [(unsigned char)(cookie[2])] = ‘9’ = 39

PHASE3\_CODEBOOK [(unsigned char)(cookie[3])] = ‘0’ = 30

PHASE3\_CODEBOOK [(unsigned char)(cookie[4])] = ‘2’ = 32

PHASE3\_CODEBOOK [(unsigned char)(cookie[5])] = ‘0’ = 30

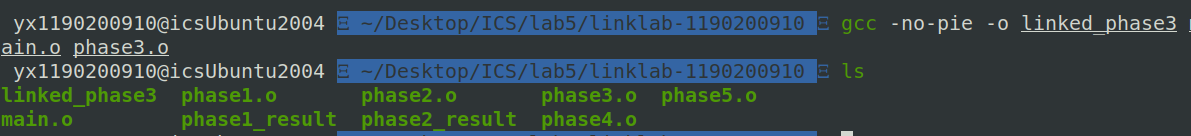
PHASE3\_CODEBOOK [(unsigned char)(cookie[6])] = ‘0’ = 30

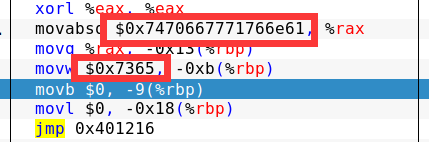
PHASE3\_CODEBOOK [(unsigned char)(cookie[7])] = ‘9’ = 39

PHASE3\_CODEBOOK [(unsigned char)(cookie[8])] = ‘1’ = 31

PHASE3\_CODEBOOK [(unsigned char)(cookie[9])] = ‘0’ = 30

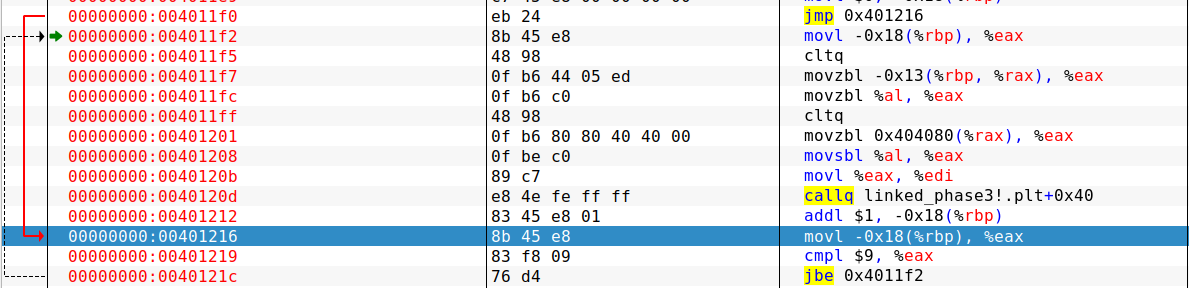
为了构造PHASE3\_CODEBOOK数组，我们必须知道cookie指向的那个数组中的内容。因此我们先用原始的phase3.o和main.o链接，并用EDB调试这个程序。





在do\_phase的汇编中看到两处奇怪的立即数，两个立即数放在了从rbp-0x13开始的连续10个字节内。而我们的学号就是10个字节的，因此怀疑这就是do\_phase函数一开始给cookie数组所赋的值。10个字节数分别是：0x74, 0x70, 0x66, 0x77, 0x71, 0x76, 0x6e, 0x61, 0x73, 0x65。

通过do\_phase下面的汇编代码分析，这一处就是for循环的代码。



在rbp-0x18处保存了循环变量i，如果i<=9就会以i为变址，从rbp-0x13处进行寻址，并把数组寻址的结果放到eax中，这也就印证了我们上面的推测。因此现在可以确定，cookie数组中的元素为：{0x74, 0x70, 0x66, 0x77, 0x71, 0x76, 0x6e, 0x61, 0x73, 0x65}。

因此我们需要构造的PHASE3\_CODEBOOK数组必须满足：

PHASE3\_CODEBOOK[0x74]=’1’=0x31

PHASE3\_CODEBOOK[0x70]=’1’=0x31

PHASE3\_CODEBOOK[0x66]=’9’=0x39

PHASE3\_CODEBOOK[0x77]=’0’=0x30

PHASE3\_CODEBOOK[0x71]=’2’=0x32

PHASE3\_CODEBOOK[0x76]=’0’=0x30

PHASE3\_CODEBOOK[0x6e]=’0’=0x30

PHASE3\_CODEBOOK[0x61]=’9’=0x39

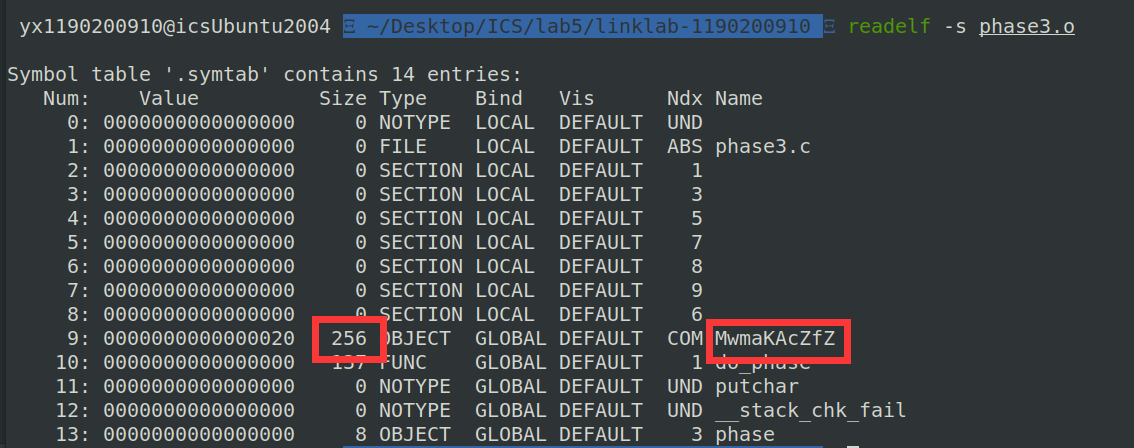
PHASE3\_CODEBOOK[0x73]=’1’=0x31

PHASE3\_CODEBOOK[0x65]=’0’=0x30

于是，定义数组：

char PHASE3\_CODEBOOK[256] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 57, 0, 0, 0, 48, 57, 0, 0, 0, 0, 0, 0, 0, 48, 0, 49, 50, 0, 49, 49, 0, 48, 48, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

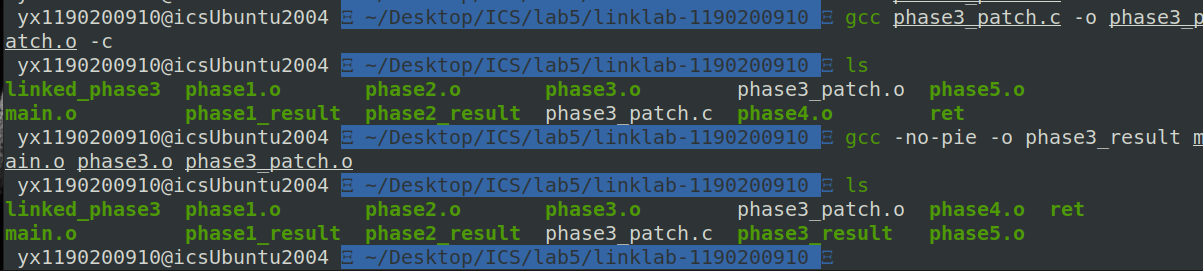
利用readelf -s查看phase3.o的符号表：



其中MwmaKAcZfZ即为PHASE3\_CODEBOOK的名字。因此我们在phase3\_patch.c文件中也定义一个同名的数组，并且把上面计算出的数组元素值给赋进去：



然后先编译phase3\_patch（不链接），然后将main.o, phase3.o, phase3\_patch.o三个文件一起链接。



最后运行生成的程序：



发现结果不是正确的学号。顺序发生了错误，前8个数字倒序了，后2个数字也倒序了。

回过头来找错误原因，发现是分析do\_phase代码时，给cookie数组赋初值时，应该是小端序存储数据，而我跟随思维定势当作大端序，从低地址到高地址赋值了。其中第一个立即数是0x7470667771766e61，转换成小端字节，从低地址到高地址就是：0x61, 0x6e, 0x76, 0x71, 0x77, 0x66, 0x70, 0x74；第二个立即数是0x7365，转化为小端字节，从低地址到高地址为0x65, 0x73。

于是修改PHASE3\_CODEBOOK数组中的赋值顺序：

PHASE3\_CODEBOOK[0x61]=0x31;

PHASE3\_CODEBOOK[0x6e]= 0x31;

PHASE3\_CODEBOOK[0x76]=0x39;

PHASE3\_CODEBOOK[0x71]=0x30;

PHASE3\_CODEBOOK[0x77]=0x32;

PHASE3\_CODEBOOK[0x66]=0x30;

PHASE3\_CODEBOOK[0x70]=0x30;

PHASE3\_CODEBOOK[0x74]=0x39;

PHASE3\_CODEBOOK[0x65]=0x31;

PHASE3\_CODEBOOK[0x73]=0x30;

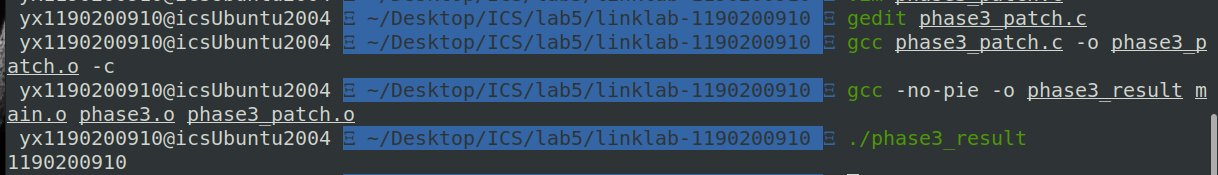
于是得到新的数组：

char PHASE3\_CODEBOOK[256] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 49, 0, 0, 0, 49, 48, 0, 0, 0, 0, 0, 0, 0, 49, 0, 48, 48, 0, 48, 57, 0, 57, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

据此对phase3\_patch.c进行修改：



修改后重新编译、链接、运行：



经过上面多步的分析、调试，修改，终于能输出正确的学号！