

实验报告

LINUX 环境和 GCC 工具链

班级：_____

学号：_____

姓名：_____

一、实验目的

- 1、熟悉 linux 操作的基本操作；
- 2、掌握 gcc 编译方法；
- 3、掌握 gdb 的调试工具使用；
- 4、掌握 objdump 反汇编工具使用；
- 5、熟悉理解反汇编程序（对照源程序与 objdump 生成的汇编程序）。

二、实验环境

ssh secure shell

三、实验概况

1、用 ssh secure shell 在 linux 环境下，用 VI 编辑书 116 页源程序：

```
#include<stdio.h>
void multstore(long, long, long*);
long mult2(long, long);
int main()
{
    long d;
    multstore(2, 3, &d);
    printf("2 * 3 --> %ld\n", d);
    return 0;
}
void multstore(long x, long y, long *dest)
{
    long t = mult2(x, y);
    *dest = t;
}
long mult2(long a, long b)
{
    long s = a * b;
    return s;
}
```

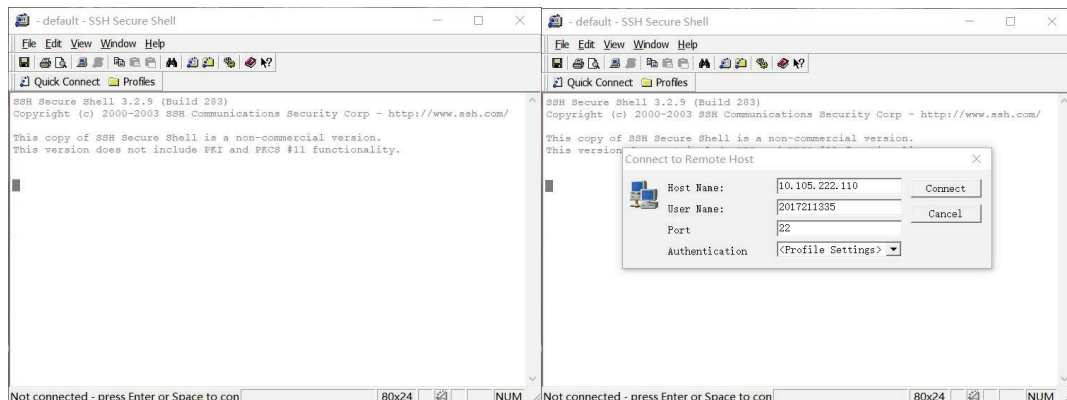
在 visualstudio 中编写好的源代码：

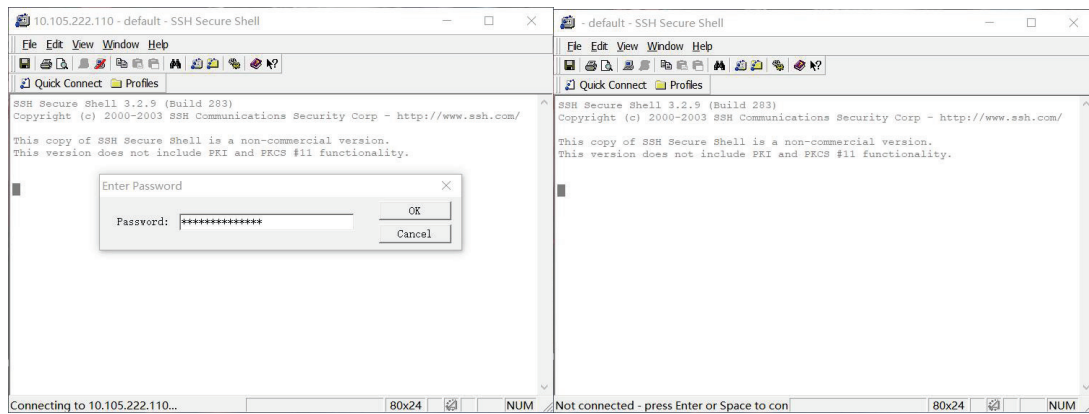


- 2、采用 gcc 编译该程序，分别采用 -o 和 -O 参数，并比较两者性能。
- 3、采用 gdb 进行调试，让程序运行到 mult2 函数的第一条语句。
- 4、运用 objdump 工具生成汇编程序。

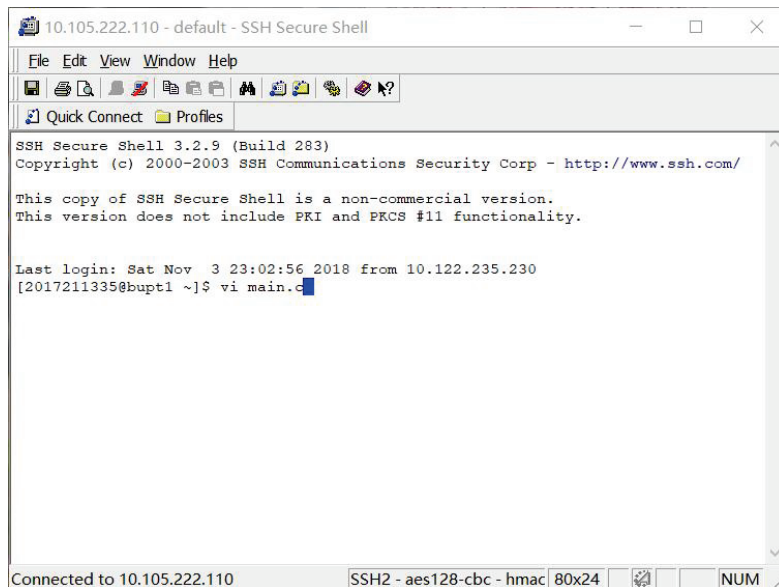
四、实验步骤

- 1、打开 ssh secure shell，登录到服务器。

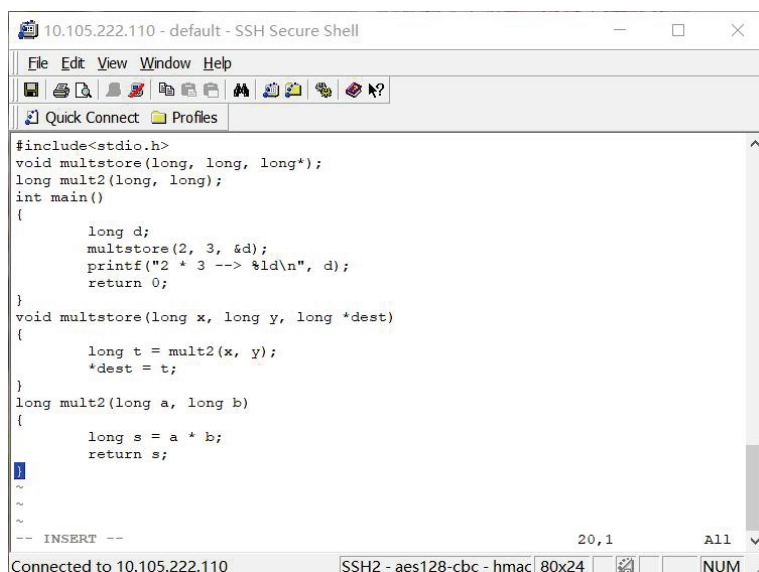




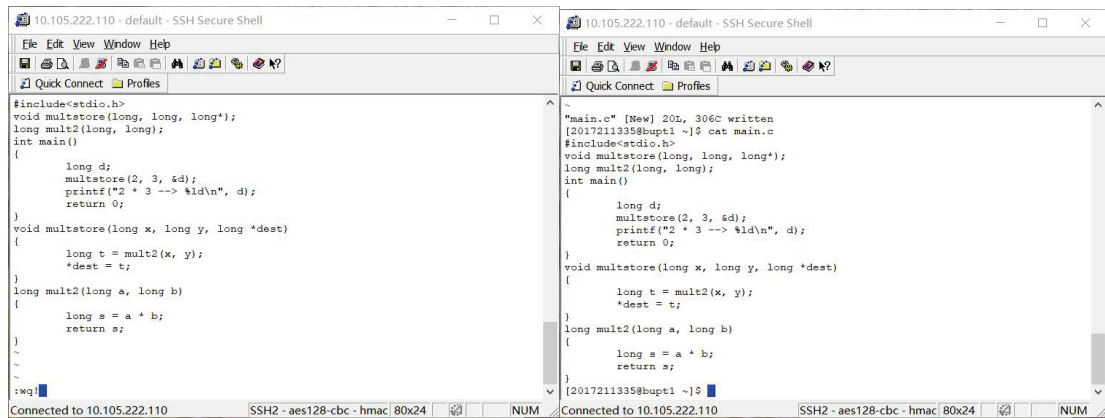
2、进入VI，新建文件 mian.c。



3、进入插入模式，编写书 116 页源程序。



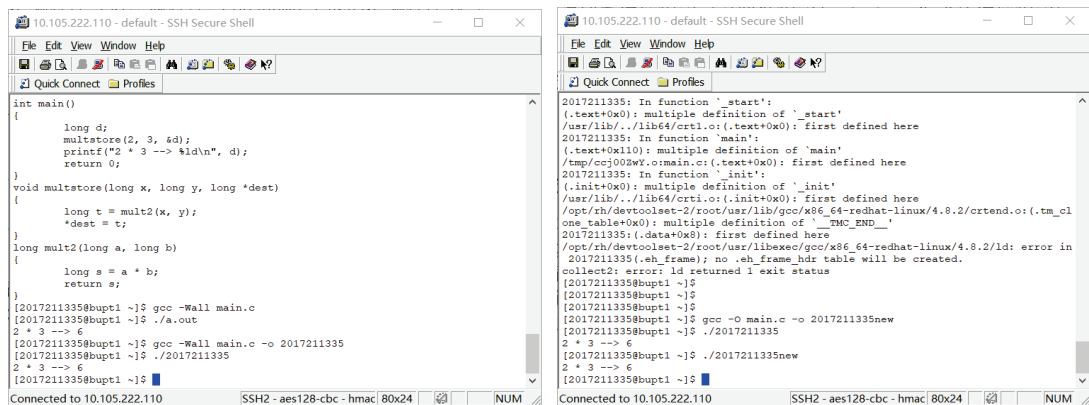
4、保存并退出。



```
#include<stdio.h>
void multstore(long, long, long*);
long mult2(long, long);
int main()
{
    long d;
    multstore(2, 3, &d);
    printf("2 * 3 --> %ld\n", d);
    return 0;
}
void multstore(long x, long y, long *dest)
{
    long t = mult2(x, y);
    *dest = t;
}
long mult2(long a, long b)
{
    long s = a * b;
    return s;
}
~
~
~
:wq
```

```
"main.c" [New] 20L, 306C written
[2017211335@bupt1 ~]$ cat main.c
#include<stdio.h>
void multstore(long, long, long*);
long mult2(long, long);
int main()
{
    long d;
    multstore(2, 3, &d);
    printf("2 * 3 --> %ld\n", d);
    return 0;
}
void multstore(long x, long y, long *dest)
{
    long t = mult2(x, y);
    *dest = t;
}
long mult2(long a, long b)
{
    long s = a * b;
    return s;
}
[2017211335@bupt1 ~]$
```

6、采用 gcc 编译该程序，分别用 -o 和 -O 参数，比较性能。

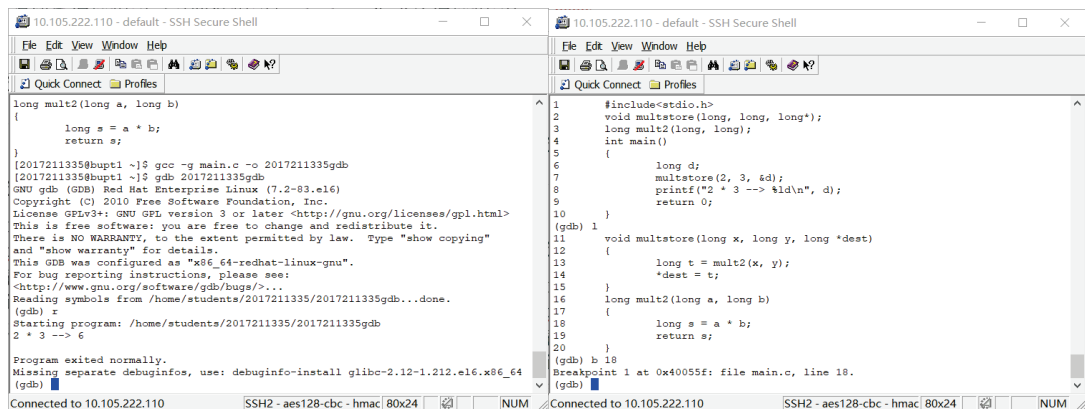


```
int main()
{
    long d;
    multstore(2, 3, &d);
    printf("2 * 3 --> %ld\n", d);
    return 0;
}
void multstore(long x, long y, long *dest)
{
    long t = mult2(x, y);
    *dest = t;
}
long mult2(long a, long b)
{
    long s = a * b;
    return s;
}
[2017211335@bupt1 ~]$ gcc -Wall main.c
[2017211335@bupt1 ~]$ ./a.out
2 * 3 --> 6
[2017211335@bupt1 ~]$ gcc -Wall main.c -o 2017211335
[2017211335@bupt1 ~]$ ./2017211335
2 * 3 --> 6
[2017211335@bupt1 ~]$
```

```
2017211335: In function '_start':
(.text+0x0): multiple definition of '_start'
/usr/lib/./lib64/crt1.o: (.text+0x0): first defined here
2017211335: In function 'main':
(.text+0x10): multiple definition of 'main'
/tmp/ccj002wY.o:main.o: (.text+0x0): first defined here
2017211335: In function '_init':
(.init+0x0): multiple definition of '_init'
/usr/lib/./lib64/crti.o: (.init+0x0): first defined here
/opt/rh/devtoolset-2/root/usr/lib/gcc/x86_64-redhat-linux/4.8.2/crtend.o: (.tm_cl
one_table+0x0): multiple definition of '__TMC_END__'
2017211335: (.data+0x8): first defined here
/opt/rh/devtoolset-2/root/usr/libexec/gcc/x86_64-redhat-linux/4.8.2/ld: error in
2017211335(.eh_frame): no .eh_frame_hdr table will be created.
collect2: error: ld returned 1 exit status
[2017211335@bupt1 ~]$
[2017211335@bupt1 ~]$ gcc -O main.c -o 2017211335new
[2017211335@bupt1 ~]$ ./2017211335new
2 * 3 --> 6
[2017211335@bupt1 ~]$
```

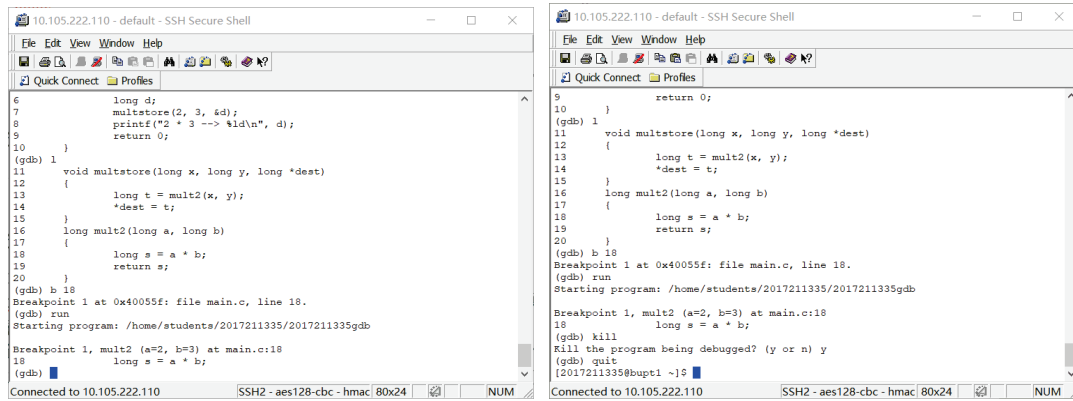
显然是 -O 性能更好。

7、采用 gdb 进行调试，让程序运行到 mult2 函数的第一条语句，并退出 gdb。



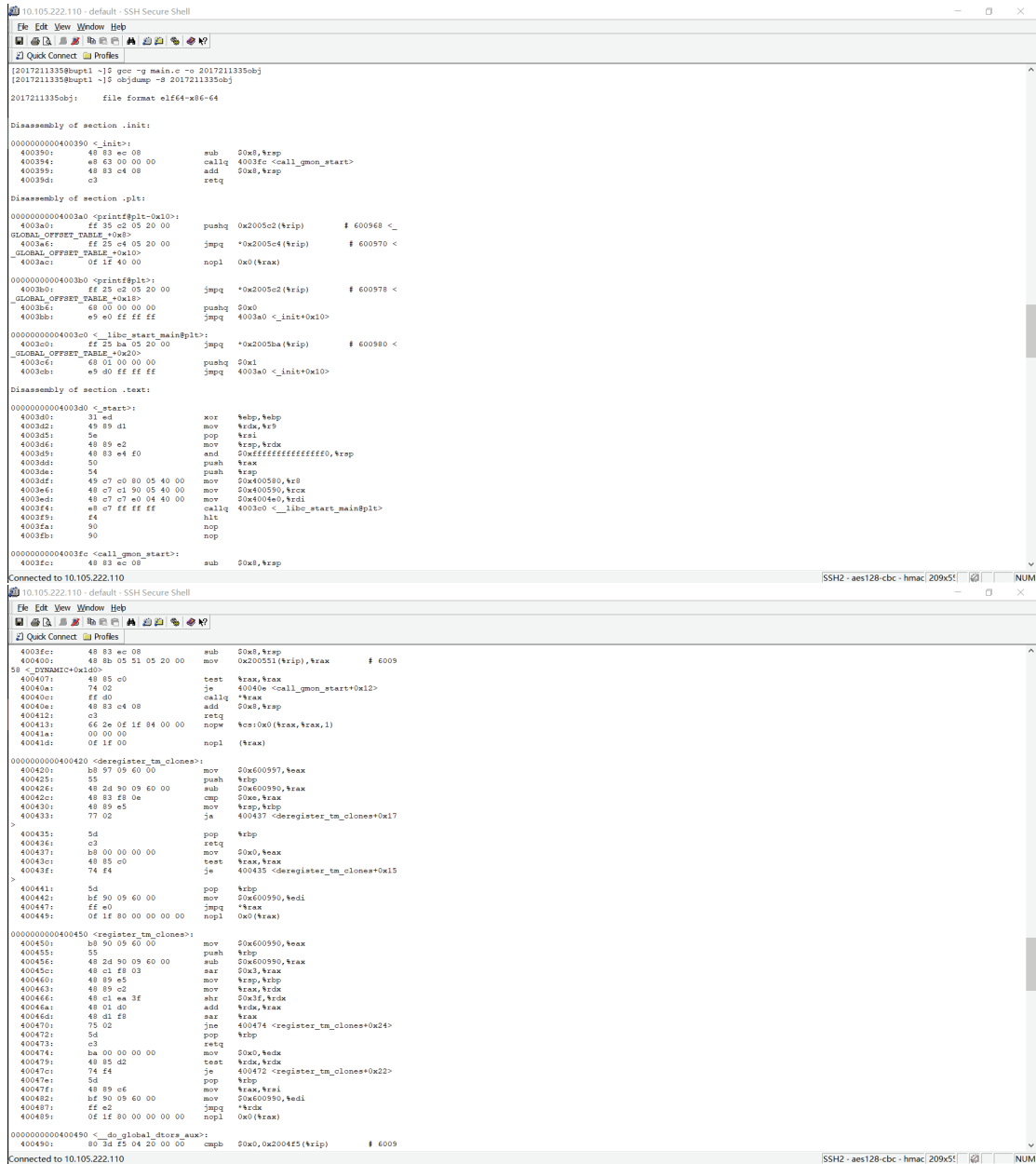
```
long mult2(long a, long b)
{
    long s = a * b;
    return s;
}
[2017211335@bupt1 ~]$ gcc -g main.c -o 2017211335gdb
[2017211335@bupt1 ~]$ gdb 2017211335gdb
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-83.el6)
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/students/2017211335/2017211335gdb...done.
(gdb) r
Starting program: /home/students/2017211335/2017211335gdb
2 * 3 --> 6
Program exited normally.
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.212.el6.x86_64
(gdb)
```

```
1 #include<stdio.h>
2 void multstore(long, long, long*);
3 long mult2(long, long);
4 int main()
5 {
6     long d;
7     multstore(2, 3, &d);
8     printf("2 * 3 --> %ld\n", d);
9     return 0;
10 }
(gdb) l
11 void multstore(long x, long y, long *dest)
12 {
13     long t = mult2(x, y);
14     *dest = t;
15 }
16 long mult2(long a, long b)
17 {
18     long s = a * b;
19     return s;
20 }
(gdb) b 18
Breakpoint 1 at 0x40055f: file main.c, line 18.
(gdb)
```



8、运用 objdump 工具生成汇编程序。

整个程序汇编代码：



```
10.105.222.110 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

400490: 80 3d f5 04 20 00 00 cmpb $0x0,0x2004f5(%rip) # 6009
8c <edata>: 75 11 jne 4004aa <__do_global_ctors_aux+0x1>
a^ 400495: 55 push %rbp
40049a: 48 89 e5 mov %rsp,%rbp
40049d: e9 7a ff ff ff callq 400420 <deregister_tm_clones>
4004a2: 5d pop %rbp
4004a3: c6 05 e2 04 20 00 01 movb $0x1,0x2004e2(%rip) # 6009
8c <edata>: f3 c3 repz retq
4004ac: 0f 1f 40 00 nopl 0x0(%rax)

00000000004004b0 <frame_dummy>:
4004b0: 48 83 3d c0 02 20 00 cmpq $0x0,0x2002c0(%rip) # 6007
80 <__JCR_END__>:
4004b7: 00
4004b8: 74 1e je 4004d8 <frame_dummy+0x28>
4004ba: b8 00 00 00 00 mov $0x0,%eax
4004bd: 48 85 c0 test %rax,%rax
4004c0: 74 1e je 4004d8 <frame_dummy+0x28>
4004c4: 55 push %rbp
4004c5: hf 80 07 60 00 mov $0x600760,%edi
4004ca: 48 89 e5 mov %rsp,%rbp
4004cd: ff d0 callq *%rax
4004cf: 5d pop %rbp
4004d0: e9 7b ff ff ff jmpq 400450 <register_tm_clones>
4004d3: 0f 1f 00 nopl (%rax)
4004d8: e9 73 ff ff ff jmpq 400450 <register_tm_clones>
4004dd: 0f 1f 00 nopl (%rax)

00000000004004e0 <main>:
#include<stdio.h>
void multstore(long, long, long*);
long mult2(long, long);
int main()
{
4004e0: 55 push %rbp
4004e1: 48 89 e5 mov %rsp,%rbp
4004e4: 48 83 ec 10 sub $0x10,%rsp
long d;
4004e8: 48 8d 45 f8 lea -0x8(%rbp),%rax
4004ec: 48 85 c5 mov %rax,%rdx
4004ee: be 03 00 00 00 mov $0x3,%esi
4004f4: hf 02 00 00 00 mov $0x2,%edi
4004f9: e9 1d 00 00 00 callq 40051b <multstore>
printf("2 * 3 -> %ld\n", d);
4004fe: 48 8d 45 f8 lea -0x8(%rbp),%rax
400502: 48 89 c6 mov %rax,%rax
400505: bf 38 06 40 00 mov $0x400638,%edi
40050a: bf 00 00 00 00 mov $0x0,%eax
40050f: e9 9c ff ff ff callq 4003b0 <printf@plt>
return 0;
400514: b8 00 00 00 00 mov $0x0,%eax
}
400514: b8 00 00 00 00 mov $0x0,%eax
400519: c9 leaveq
40051a: c3 retq

000000000040051b <multstore>:
void multstore(long x, long y, long *dest)
{
40051b: 55 push %rbp
40051c: 48 89 e5 mov %rsp,%rbp
40051f: 48 83 ec 30 sub $0x30,%rsp
400523: 48 89 7d e0 mov %rdi,-0x18(%rbp)
400527: 48 89 75 e0 mov %rsi,-0x20(%rbp)
40052b: 48 89 55 d8 mov %rdx,-0x28(%rbp)
long z = mult2(x, y);
40052f: 48 8b 55 e0 mov -0x20(%rbp),%rdx
400533: 48 8b 45 e0 mov -0x18(%rbp),%rax
400537: 48 89 d6 mov %rdx,%rax
40053a: 48 89 c7 mov %rax,%rdi
40053d: e9 11 00 00 00 callq 400553 <mult2>
400542: 48 89 45 f8 mov %rax,-0x8(%rbp)
*dest = z;
400546: 48 8b 45 d8 mov -0x28(%rbp),%rax
40054a: 48 8b 55 f8 mov -0x8(%rbp),%rdx
40054e: 48 89 10 mov %rdx,%rax
}
400551: c9 leaveq
400552: c3 retq

0000000000400553 <mult2>:
long mult2(long a, long b)
{
400553: 55 push %rbp
400554: 48 89 e5 mov %rsp,%rbp
400557: 48 89 7d e0 mov %rdi,-0x18(%rbp)
40055b: 48 89 75 e0 mov %rsi,-0x20(%rbp)
long s = a * b;
40055f: 48 8b 45 e0 mov -0x18(%rbp),%rax
400563: 48 0f af 45 e0 imul -0x20(%rbp),%rax
400568: 48 89 45 f8 mov %rax,-0x8(%rbp)
return s;
40056c: 48 8b 45 f8 mov -0x8(%rbp),%rax
}
400570: 5d pop %rbp
400571: c3 retq
400572: 66 2a 0f 1f 84 00 00 nopw %cs:0xa(%rax,%rax,1)
400579: 00 00 00 nopl 0x0(%rax)
40057c: 0f 1f 40 00 nopl 0x0(%rax)

0000000000400580 <__libc_csu_fini>:
400580: f3 c3 repz retq
400582: 66 66 66 66 2e 0f data32 data32 data32 data32 nopw %cs:0xa
(%rax,%rax,1)
400589: 1f 84 00 00 00 00 00 nopl 0x0(%rax,%rax,1)

Connected to 10.105.222.110
SSH2 - aes128-cbc - hmac 209x5t! NUM
```

