Local or global 实验报告

缪晨露

3170102668

1. 实验问题

Many language rules are checked by the compiler, and it is possible to bypass the rules using assembly language after compilation. Consider the following C program

```
include<stdio.h>
int x=3;

int main(void)
{
  int x=5;
  printf("x = %d\n", x);
}
```

1. Compile the program and generate its assembly code.
2. Understand the assembly code and modify it to let the program print the global variable `x` instead of the local variable `x`.

2. 实验过程

2.1 生成汇编

对上述代码进行编译得到汇编文件

```
gcc -S local.cpp
```

得到文件local.s

local.s内容为

```
    .file   "local.cpp"
    .globl  x
    .data
```

```
    .align 4
    .type   x, @object
    .size   x, 4
x:
    .long   3
    .section    .rodata
.LC0:
    .string "x = %d\n"
    .text
    .globl  main
    .type   main, @function
main:
.LFB0:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    subq    $16, %rsp
    movl    $5, -4(%rbp)
    movl    -4(%rbp), %eax
    movl    %eax, %esi
    movl    $.LC0, %edi
    movl    $0, %eax
    call    printf
    movl    $0, %eax
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size   main, .-main
    .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609"
    .section    .note.GNU-stack,"",@progbits
```

`gcc -c local.s` 得到local.o文件

```
jones@ubuntu:~/Desktop/csapp/hw3$ objdump -d local.o

local.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   48 83 ec 10             sub    $0x10,%rsp
   8:   c7 45 fc 05 00 00 00    movl   $0x5,-0x4(%rbp)
   f:   8b 45 fc                mov    -0x4(%rbp),%eax
  12:   89 c6                   mov    %eax,%esi
```

```
  14:   bf 00 00 00 00          mov    $0x0,%edi
  19:   b8 00 00 00 00          mov    $0x0,%eax
  1e:   e8 00 00 00 00          callq  23 <main+0x23>
  23:   b8 00 00 00 00          mov    $0x0,%eax
  28:   c9                      leaveq
  29:   c3                      retq
```

下面代码用来保护堆栈

```
push   %rbp
mov    %rsp,%rbp
```

然后腾出栈空间

```
sub    $0x10,%rsp
```

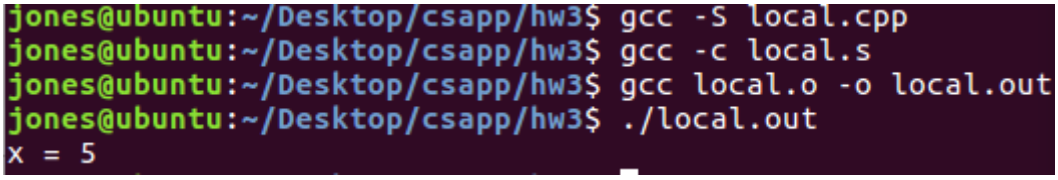
为调用printf函数，需要准备相关参数

```
movl   $0x5,-0x4(%rbp)
mov    -0x4(%rbp),%eax
mov    %eax,%esi
mov    $0x0,%edi
mov    $0x0,%eax
callq  23 <main+0x23>
```

从这段代码中可以看出：

调用printf函数之前，将参数放入`%edi`，`%esi`寄存器

`%esi`中放的是local variable x的值



2.2 输出global变量

修改local.s的代码

```
movl    $5, -4(%rbp)
movl    -4(%rbp), %eax
```

为

```
movl    x(%rip), %eax
```

global.s部分代码如下图所示

```
x:
        .long   3
        .section        .rodata
.LC0:
        .string "x = %d\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    x(%rip), %eax
        movl    %eax, %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
```

再来看一下代码的运行情况

```
jones@ubuntu:~/Desktop/csapp/hw3$ objdump -d global.o

global.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   48 83 ec 10             sub    $0x10,%rsp
   8:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # e <main+0xe>
   e:   89 c6                   mov    %eax,%esi
  10:   bf 00 00 00 00          mov    $0x0,%edi
  15:   b8 00 00 00 00          mov    $0x0,%eax
  1a:   e8 00 00 00 00          callq  1f <main+0x1f>
  1f:   b8 00 00 00 00          mov    $0x0,%eax
  24:   c9                      leaveq
  25:   c3                      retq
```

```
jones@ubuntu:~/Desktop/csapp/hw3$ gcc local.o -o local.out
jones@ubuntu:~/Desktop/csapp/hw3$ ./local.out
x = 3
```