ICS第三章

【结构与联合】

15. 在x86-64、Linux操作系统下有如下C定义：

|  |
| --- |
| struct A {  char CC1[6];  int II1;  long LL1;  char CC2[10];  long LL2;  int II2;  }; |

(1) sizeof(A) = 字节。

(2) 将A重排后，令结构体尽可能小，那么得到的新的结构体大小为 字节。

16. 在x86-64、LINUX操作系统下，考虑如下的C定义：

|  |
| --- |
| typedef union {  char c[7];  short h;  } union\_e;  typedef struct {  char d[3];  union\_e u;  int i;  } struct\_e;  struct\_e s; |

回答如下问题：

(1) s.u.c的首地址相对于s的首地址的偏移量是\_\_\_\_\_\_\_\_字节。

(2) sizeof(union\_e) = \_\_\_\_\_\_\_\_字节。

(3) s.i的首地址相对于s的首地址的偏移量是\_\_\_\_\_\_\_\_字节。

(4) sizeof(struct\_e) = \_\_\_\_\_\_\_\_字节。

(5) 若只将i的类型改成short，那么sizeof(struct\_e) = \_\_\_\_\_\_\_\_字节。

(6) 若只将h的类型改成int，那么sizeof(union\_e) = \_\_\_\_\_\_\_\_字节。

(7) 若将i的类型改成short、将h的类型改成int，那么sizeof(union\_e) = \_\_\_\_\_\_\_\_字节，sizeof(struct\_e) = \_\_\_\_\_\_\_\_字节。

(8) 若只将short h的定义删除，那么(1)~(4)问的答案分别是\_\_\_，\_\_\_，\_\_\_，\_\_\_。

【调试工具】

17. 写出使用gcc编译源代码lab.c、生成可执行文件lab、采用二级编译优化的命令。

18. 写出使用gcc编译源代码foo.c、生成汇编语言文件foo.s的命令

19. 将可执行文件bar逆向工程为汇编代码的工具是( )。

A. gcc B. gdb C. objdump D. hexedit E. gedit

20. gdb中，单步指令执行的命令是( )。

A. r B. b C. p D. finish E. si F. disas

【综合】

21. 以下提供了一段代码的C语言、汇编语言以及运行到某一时刻栈的情况

汇编:

|  |
| --- |
| 0000000000400596 <func>:  400596: sub $0x28,%rsp  40059a: mov %fs:0x28,%rax  4005a3: mov %rax,0x18(%rsp)  4005a8: xor %eax,%eax  4005aa: mov (%rdi),%rax  4005ad: mov 0x8(%rdi),%rdx  4005b1: cmp %rdx,%rax  4005b4: jge (1)  4005b6: mov %rdx,(%rdi)  4005b9: mov %rax,0x8(%rdi)  4005bd: mov 0x8(%rdi),%rax  4005c1: test %rax,%rax  4005c4: jne 4005cb <func+0x35>  4005c6: mov (%rdi),%rax  4005c9: jmp (2)  4005cb: mov (%rdi),%rdx  4005ce: sub %rax,%rdx  4005d1: mov %rdx,(%rsp)  4005d5: mov %rax,0x8(%rsp)  4005da: mov (3) ,%rdi  4005dd: callq 400596 <func>  4005e2: mov 0x18(%rsp),%rcx  4005e7: xor (4) ,%rcx  4005f0: (5) 4005f7 <func+0x61>  4005f2: callq 400460 <\_\_stack\_chk\_fail@plt>  4005f7: add (6) ,%rsp  4005fb: retq  00000000004005fc <main>:  4005fc: sub $0x28,%rsp  400600: mov %fs:0x28,%rax  400609: mov %rax,0x18(%rsp)  40060e: xor %eax,%eax  400610: movq 0x69,(%rsp)  400618: movq 0xfc,0x8(%rsp)  400621: mov %rsp,%rdi  400624: callq 400596 <func>  400629: mov %rax,%rsi  40062c: mov $0x4006e4,%edi  400631: mov $0x0,%eax  400636: callq 400470 <printf@plt>  40063b: mov 0x18(%rsp),%rdx  400640: xor (4) ,%rdx  400649: (5) 400650 <main+0x54>  40064b: callq 400460 <\_\_stack\_chk\_fail@plt>  400650: mov $0x0,%eax  400655: add (6) ,%rsp  400659: retq |

C语言与堆栈:

|  |  |
| --- | --- |
| typedef struct{  long a;  long b;  } pair\_type;  long func(pair\_type \*p) {  if (p -> a < p -> b) {  long temp = p -> a;  p -> a = p -> b;  p -> b = temp;  }  if ((7) ) {  return p -> a;  }  pair\_type np;  np.a = (8) ;  np.b = (9) ;  return func(&np);  }  int main(int argc, char\* argv[]) {  pair\_type np;  np.a = (10) ;  np.b = (11) ;  printf("%ld", func(&np));  return 0;  } | ...... |
| 0x0000000000000000 |
| 0xc76d5add7bbeaa00 |
| 0x00007fffffffdf60 |
| (a) |
| (b) |
| 0x0000000000400629 |
| (c) |
| (d) |
| 0x0000000000000001 |
| 0x0000000000000069 |
| 0x0000000000000093 |
| (e) |
| 0x00000000ff000000 |
| (f) |
| 0x0000000000000000 |
| (g) |
| (h) |
| (i) |
| 0x0000000000000000 |
| (j) |
| (k) |
| 0x000000000000002a |
| 0x000000000000003f |
| 0x00000000004005e2 |
| （栈顶）[低地址] |

一些可能用到的字符的ASCII码表:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 换行 | 空格 | " | % | ( | ) | , | 0 | A | a |
| 0x0a | 0x20 | 0x22 | 0x25 | 0x28 | 0x29 | 0x2c | 0x30 | 0x41 | 0x61 |

I. gdb下使用命令x/4b 0x4006e4后（即查看0x4006e4开始的4个字节，用16进制表示）得到的输出结果是

0x4006e4: 0x 0x 0x 0x

II. 互相翻译C语言代码和汇编代码，补充缺失的空格（标号相同的为同一格）。

(1) <func+ >   
(2) <func+ >   
(3)   
(4)   
(5)   
(6)   
(7)   
(8)   
(9)   
(10)   
(11)

III. 补充栈的内容。使用16进制，可以不写前导多余的0；对于给定已知条件后仍无法确定的值，填写“不确定”；已知程序运行过程中寄存器%fs的值没有改变。

(a)   
(b)   
(c)   
(d)   
(e)   
(f)   
(g)   
(h)   
(i)   
(j)   
(k)

IV. 程序运行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。