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。

【答】56，40；CC1：0；II1：8，必须 4 字节对齐；LL1：16，必须 8 字节对齐；CC2：24；LL2：40，必须 8 字节对齐；II2：48。基本元素最大的为 long，因此 sizeof(A)是 56。重排顺序：LL1 LL2 II1 II2 CC1 CC2，刚好没有空白空间，得到的大小为 8+8+4+4+10+6=40 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___，___，___，___。

【答】

4; 8; 12; 16; 14; 8; 8, 16; 3, 7, 12, 16。具体解释如下：

(1)~(4)：

<table>
<tr><td>d1</td><td>d2</td><td>d3</td><td></td><td colspan="4">h</td><td></td><td></td><td></td><td></td><td></td><td colspan="4">i</td></tr>
<tr><td></td><td></td><td></td><td></td><td>c1</td><td>c2</td><td>c3</td><td>c4</td><td>c5</td><td>c6</td><td>c7</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

(5)：

<table>
<tr><td>d1</td><td>d2</td><td>d3</td><td></td><td colspan="4">h</td><td></td><td></td><td></td><td></td><td></td><td colspan="2">i</td></tr>
<tr><td></td><td></td><td></td><td></td><td>c1</td><td>c2</td><td>c3</td><td>c4</td><td>c5</td><td>c6</td><td>c7</td><td></td><td></td><td></td></tr>
</table>

(6)：

<table>
<tr><td>d1</td><td>d2</td><td>d3</td><td></td><td colspan="7">h</td><td></td><td></td><td colspan="4">i</td></tr>
<tr><td></td><td></td><td></td><td></td><td>c1</td><td>c2</td><td>c3</td><td>c4</td><td>c5</td><td>c6</td><td>c7</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

(7)：

<table>
<tr><td>d1</td><td>d2</td><td>d3</td><td></td><td colspan="7">h</td><td></td><td></td><td></td><td colspan="2">i</td></tr>
<tr><td></td><td></td><td></td><td></td><td>c1</td><td>c2</td><td>c3</td><td>c4</td><td>c5</td><td>c6</td><td>c7</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

对于(7)，虽然和(5)很像，并且 sizeof(union_e)也没变，但是实际上对齐要求的是所有基本元素都要对齐。所以在考虑 sizeof 的时候，不妨假设开辟一个包含两个元素的结构体数组，检查一下第二个结构体是否对其了所有的基本元素。

(8)：

<table>
<tr><td>d1</td><td>d2</td><td>d3</td><td>c1</td><td>c2</td><td>c3</td><td>c4</td><td>c5</td><td>c6</td><td>c7</td><td></td><td></td><td colspan="4">i</td></tr>
</table>

【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。

【答】gcc lab.c -o lab -O2

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

【答】gcc foo.c -S

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

【答】C

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

【答】E

【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

回答问题：

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_____。

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

汇编代码：

```
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    4005bd <func+0x27>
  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    4005e2 <func+0x4c>
  4005cb:  mov    (%rdi),%rdx
  4005ce:  sub    %rax,%rdx
  4005d1:  mov    %rdx,(%rsp)
  4005d5:  mov    %rax,0x8(%rsp)
  4005da:  mov    %rsp,%rdi
  4005dd:  callq  400596 <func>
  4005e2:  mov    0x18(%rsp),%rcx
  4005e7:  xor    %fs:0x28,%rcx
  4005f0:  je     4005f7 <func+0x61>
  4005f2:  callq  400460 <__stack_chk_fail@plt>
  4005f7:  add    $0x28,%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    %fs:0x28,%rdx
  400649:  je     400650 <main+0x54>
  40064b:  callq  400460 <__stack_chk_fail@plt>
  400650:  mov    $0x0,%eax
  400655:  add    $0x28,%rsp
  400659:  retq
```

C 语言与 stack 的情况：

```
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 (p -> b == 0) {
        return p -> a;
    }
    pair_type np;
    np.a = p -> a - p -> b;
    np.b = p -> b;
    return func(&np);
}
int main(int argc, char* argv[]) {
    pair_type np;
    np.a = 105;
    np.b = 252;
    printf("%ld", func(&np));
    return 0;
}
```

```
......
0x0000000000000000(u)
0xc76d5add7bbeaa00
0x00007fffffffdf60(u?)
0x0000000000000069
0x00000000000000fc
0x0000000000400629
0x0000000000000000(u)
0xc76d5add7bbeaa00
0x0000000000000001(u)
0x0000000000000069
0x0000000000000093
0x00000000004005e2
0x00000000ff000000(u)
0xc76d5add7bbeaa00
0x0000000000000000(u)
0x000000000000002a
0x0000000000000069
0x00000000004005e2
0x0000000000000000(u)
0xc76d5add7bbeaa00
0x000000ff00000000(u)
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

回答问题：

1．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: 0x25 0x6c 0x64 0x00

2．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) 4005bd <func+0x27>
(2) 4005e2 <func+0x4c>
(3) %rsp
(4) %fs:0x28
(5) je
(6) $0x28
(7) p -> b == 0
(8) p -> a - p -> b
(9) p -> b
(10) 105
(11) 252

3．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) 0x0000000000000069
(b) 0x00000000000000fc
(c) 不确定
(d) 0xc76d5add7bbeaa00
(e) 0x00000000004005e2
(f) 0xc76d5add7bbeaa00
(g) 0x000000000000002a
(h) 0x0000000000000069
(i) 0x00000000004005e2
(j) 0xc76d5add7bbeaa00
(k) 不确定

4．程序输出结果是？

21

【注意】
1 最后一空为 0x00，因为要用\0 作字符串结尾
2(1)(2)较难
2(10)(11)不能颠倒，因为汇编如此写
3(a)(b)两空不能颠倒，因为 func 函数会修改内存的值
程序的作用是用（类似于）更相减损术求最大公约数