```
   //// 移动到用户模式运行。
   // 该函数利用 iret 指令实现从内核模式移动到初始任务 0 中去执行。
 1 #define move_to_user_mode() \
 2 __asm__ ("movl %%esp,%%eax\n\t" \          // 保存堆栈指针 esp 到 eax 寄存器中。
 3         "pushl $0x17\n\t" \                 // 首先将堆栈段选择符(SS)入栈。
 4         "pushl %%eax\n\t" \                 // 然后将保存的堆栈指针值(esp)入栈。
 5         "pushfl\n\t" \                      // 将标志寄存器(eflags)内容入栈。
 6         "pushl $0x0f\n\t" \                 // 将 Task0 代码段选择符(cs)入栈。
 7         "pushl $1f\n\t" \                   // 将下面标号 1 的偏移地址(eip)入栈。
 8         "iret\n" \                          // 执行中断返回指令，则会跳转到下面标号 1 处。
 9         "1:\tmovl $0x17,%%eax\n\t" \        // 此时开始执行任务 0，
10         "movw %%ax,%%ds\n\t" \              // 初始化段寄存器指向本局部表的数据段。
11         "movw %%ax,%%es\n\t" \
12         "movw %%ax,%%fs\n\t" \
13         "movw %%ax,%%gs" \
14         :::"ax")
15
16 #define sti() __asm__ ("sti"::)             // 开中断嵌入汇编宏函数。
17 #define cli() __asm__ ("cli"::)             // 关中断。
18 #define nop() __asm__ ("nop"::)             // 空操作。
19
20 #define iret() __asm__ ("iret"::)           // 中断返回。
21
   //// 设置门描述符宏。
   // 根据参数中的中断或异常处理过程地址 addr、门描述符类型 type 和特权级信息 dpl，设置位于
   // 地址 gate_addr 处的门描述符。（注意：下面“偏移”值是相对于内核代码或数据段来说的）。
   // 参数：gate_addr -描述符地址；type -描述符类型域值；dpl -描述符特权级；addr -偏移地址。
   // %0 - (由 dpl,type 组合成的类型标志字)；%1 - (描述符低 4 字节地址)；
   // %2 - (描述符高 4 字节地址)；%3 - edx(程序偏移地址 addr)；%4 - eax(高字中含有段选择符 0x8)。
22 #define _set_gate(gate_addr,type,dpl,addr) \
23 __asm__ ("movw %%dx,%%ax\n\t" \             // 将偏移地址低字与选择符组合成描述符低 4 字节(eax)。
24         "movw %0,%%dx\n\t" \                // 将类型标志字与偏移高字组合成描述符高 4 字节(edx)。
25         "movl %%eax,%1\n\t" \               // 分别设置门描述符的低 4 字节和高 4 字节。
26         "movl %%edx,%2" \
27         : \
28         : "i" ((short) (0x8000+(dpl<<13)+(type<<8))), \
29         "o" (*((char *) (gate_addr))), \
30         "o" (*(4+(char *) (gate_addr))), \
31         "d" ((char *) (addr)),"a" (0x00080000))
32
   //// 设置中断门函数（自动屏蔽随后的中断）。
   // 参数：n - 中断号；addr - 中断程序偏移地址。
   // &idt[n]是中断描述符表中中断号 n 对应项的偏移值；中断描述符的类型是 14，特权级是 0。
33 #define set_intr_gate(n,addr) \
34         _set_gate(&idt[n],14,0,addr)
35
   //// 设置陷阱门函数。
   // 参数：n - 中断号；addr - 中断程序偏移地址。
   // &idt[n]是中断描述符表中中断号 n 对应项的偏移值；中断描述符的类型是 15，特权级是 0。
36 #define set_trap_gate(n,addr) \
37         _set_gate(&idt[n],15,0,addr)
38
```

```
    //// 设置系统陷阱门函数。
    // 上面 set_trap_gate()设置的描述符的特权级为 0，而这里是 3。因此 set_system_gate()设置的
    // 中断处理过程能够被所有程序执行。例如单步调试、溢出出错和边界超出出错处理。
    // 参数：n - 中断号；addr - 中断程序偏移地址。
    // &idt[n]是中断描述符表中中断号 n 对应项的偏移值；中断描述符的类型是 15，特权级是 3。
39  #define set_system_gate(n,addr) \
40          _set_gate(&idt[n],15,3,addr)
41
    //// 设置段描述符函数（内核中没有用到）。
    // 参数：gate_addr -描述符地址；type -描述符中类型域值；dpl -描述符特权层值；
    // base - 段的基地址；limit - 段限长。
    // 请参见段描述符的格式。注意，这里赋值对象弄反了。43 行应该是 *((gate_addr)+1)，而
    // 49 行才是 *(gate_addr)。不过内核代码中没有用到这个宏，所以 Linus 没有察觉 :-)
42  #define _set_seg_desc(gate_addr,type,dpl,base,limit) {\
43          *(gate_addr) = ((base) & 0xff000000) | \                // 描述符低 4 字节。
44                  (((base) & 0x00ff0000)>>16) | \
45                  ((limit) & 0xf0000) | \
46                  ((dpl)<<13) | \
47                  (0x00408000) | \
48                  ((type)<<8); \
49          *((gate_addr)+1) = (((base) & 0x0000ffff)<<16) | \      // 描述符高 4 字节。
50                  ((limit) & 0x0ffff); }
51
    //// 在全局表中设置任务状态段/局部表描述符。状态段和局部表段的长度均被设置成 104 字节。
    // 参数：n - 在全局表中描述符项 n 所对应的地址；addr - 状态段/局部表所在内存的基地址。
    //       type - 描述符中的标志类型字节。
    // %0 - eax(地址 addr)；%1 - (描述符项 n 的地址)；%2 - (描述符项 n 的地址偏移 2 处)；
    // %3 - (描述符项 n 的地址偏移 4 处)；%4 - (描述符项 n 的地址偏移 5 处)；
    // %5 - (描述符项 n 的地址偏移 6 处)；%6 - (描述符项 n 的地址偏移 7 处)；
52  #define _set_tssldt_desc(n,addr,type) \
53  __asm__ ("movw $104,%1\n\t" \                   // 将 TSS（或 LDT）长度放入描述符长度域(第 0-1 字节)。
54          "movw %%ax,%2\n\t" \                    // 将基地址的低字放入描述符第 2-3 字节。
55          "rorl $16,%%eax\n\t" \                  // 将基地址高字右循环移入 ax 中（低字则进入高字处）。
56          "movb %%al,%3\n\t" \                    // 将基地址高字中低字节移入描述符第 4 字节。
57          "movb $" type ",%4\n\t" \               // 将标志类型字节移入描述符的第 5 字节。
58          "movb $0x00,%5\n\t" \                   // 描述符的第 6 字节置 0。
59          "movb %%ah,%6\n\t" \                    // 将基地址高字中高字节移入描述符第 7 字节。
60          "rorl $16,%%eax" \                      // 再右循环 16 比特，eax 恢复原值。
61          ::"a" (addr), "m" (*(n)), "m" (*(n+2)), "m" (*(n+4)), \
62           "m" (*(n+5)), "m" (*(n+6)), "m" (*(n+7)) \
63          )
64
    //// 在全局表中设置任务状态段描述符。
    // n - 是该描述符的指针；addr - 是描述符项中段的基地址值。任务状态段描述符的类型是 0x89。
65  #define set_tss_desc(n,addr) _set_tssldt_desc(((char *) (n)),addr,"0x89")
    //// 在全局表中设置局部表描述符。
    // n - 是该描述符的指针；addr - 是描述符项中段的基地址值。局部表段描述符的类型是 0x82。
66  #define set_ldt_desc(n,addr) _set_tssldt_desc(((char *) (n)),addr,"0x82")
67
```