源程序书写格式

尽管 C 语言在语法规定上对书写格式没有严格的限制，但是源程序的排版格式应当遵从多数程序员的惯例，以便于程序的维护。许多同学可能受中文单词间不需要空格的影响以及 C 语言教科书中未对书写格式提出建议，导致书写的 C 语言源程序缺少必要的空格，密密麻麻的语法元素拥挤在一起，阅读起来需要首先“断词”。按照惯例应当添加适当的空格，断开相关的“语法单词”。下面列举出书写方面应注意的几个问题。后面两页附上的源代码样例节选于 Linux 的源程序文件 ping.c，源代码样例的每行都带有行号，对书写方面应注意的问题提供了实际例子的参考行号。

附表 1　C 语言源程序书写格式建议

项目	说明	参考行号
适当的空行	根据程序的上下文，分成逻辑上的几段，段与段空行	7，12，33
语句的层次缩进	每个层次缩进 4 个空格	13~16
双目运算符	运算符两侧各加一个空格	18~20，51
三目运算符	运算符两侧各加一个空格，如： a > b ? a : b	
单目运算符	与它作用的变量之间不要加空格	9，32，65
关键字	关键字与它后面的括号之间保留一个空格	8，13
函数名	函数名与后面的括号之间不要保留空格，与关键字处理不同	24，32，79
宏名字	宏名字与函数名类似，名字与后面的括号之间不要保留空格	64，69
逗号	逗号后保留一个空格	24，28，80
分号	分号后保留一个空格，如：for (i = 0; i < n; i++)	
括号	左圆括号（或方括号）之后，右括号之前不要保留空格	18，24，64
换行	应当放在两行的内容，不要挤在一行内	59~60，65~66
行的长度	每行长度尽量控制在 80 字符内	76~78
注释	避免无聊的注释；避免程序修改了，不改注释，误导他人	
花括号	if，for，while，switch 等语句的花括号排版风格在整个文件中一致	8~11，64~71
命名	变量、函数及结构体的域，命名风格在整个文件中一致	1，49

花括号的书写风格有两种：

```
if (xxxxx) {
    yyyyyyyyyyyyyyyyyyyy
    yyyyyyyyyyyyyyy
} else {
    zzzzzzzzzzzzzzzz
    zzzzzzzzzzzzzzzz
}
```

```
if (xxxxx)
{
    yyyyyyyyyyyyyyyyyyyy
    yyyyyyyyyyyyyyy
}
else
{
    zzzzzzzzzzzzzzzz
    zzzzzzzzzzzzzzzz
}
```

第一种风格起源于早期 UNIX 的内核源代码，即 C 语言诞生的地方；第二种风格起源于 Pascal 程序员 begin/end 关键字的书写习惯向 C 语言的转变。目前，在 Microsoft 世界和 Linux 世界中，两种书写风格都很常见，第一种风格略占上风。无论选择哪种风格必须做到同一个源程序文件中一致，不要两种风格混用。

变量和函数命名方法也有两种风格。一种是匈牙利命名法（源自 Microsoft 的一位匈牙利籍天才程序员），如：SendAckFrame；另一种方法为传统的下划线分割法，如：send_ack_frame。同一个源程序文件中的命名风格应做到一致。

```c
static int in_cksum(unsigned short *buf, int sz)
{
   int nleft = sz;
   int sum = 0;
   unsigned short *w = buf;
   unsigned short ans = 0;

   while (nleft > 1) {
      sum += *w++;
      nleft -= 2;
    }

    if (nleft == 1) {
       *(unsigned char *)&ans = *(unsigned char *)w;
       sum += ans;
    }

    sum = (sum >> 16) + (sum & 0xFFFF);
    sum += (sum >> 16);
    ans = ~sum;
    return ans;
}

static void unpack(char *buf, int sz, struct sockaddr_in *from)
{
    struct icmp *icmppkt;
    struct iphdr *iphdr;
    struct timeval tv, *tp;
    int hlen, dupflag;
    unsigned long triptime;

    gettimeofday(&tv, NULL);

    /* check IP header */
    iphdr = (struct iphdr *) buf;
    hlen = iphdr->ihl << 2;

    /* discard if too short */
    if (sz < (datalen + ICMP_MINLEN))
       return;

    sz -= hlen;
    icmppkt = (struct icmp *) (buf + hlen);
```

```
44      if (icmppkt->icmp_id != myid)
45          return;           /* not our ping */
46
47      if (icmppkt->icmp_type == ICMP_ECHOREPLY) {
48          nreceived++;
49          tp = (struct timeval *) icmppkt->icmp_data;
50
51          if ((tv.tv_usec -= tp->tv_usec) < 0) {
52              tv.tv_sec--;
53              tv.tv_usec += 1000000;
54          }
55          tv.tv_sec -= tp->tv_sec;
56
57          triptime = tv.tv_sec * 10000 + (tv.tv_usec / 100);
58          tsum += triptime;
59          if (triptime < tmin)
60              tmin = triptime;
61          if (triptime > tmax)
62              tmax = triptime;
63
64          if (TST(icmppkt->icmp_seq % MAX_DUP_CHK)) {
65              nrepeats++;
66              nreceived--;
67              dupflag = 1;
68          } else {
69              SET(icmppkt->icmp_seq % MAX_DUP_CHK);
70              dupflag = 0;
71          }
72
73          if (options & O_QUIET)
74              return;
75
76          printf("%d bytes from %s: icmp_seq=%u", sz,
77                  inet_ntoa(*(struct in_addr *)&from->sin_addr.s_addr),
78                  icmppkt->icmp_seq);
79          printf(" ttl=%d", iphdr->ttl);
80          printf(" time=%lu.%lu ms", triptime / 10, triptime % 10);
81          if (dupflag)
82              printf(" (DUP!)");
83          printf("\n");
84      }
85  }
```