奔跑吧 LINUX社区

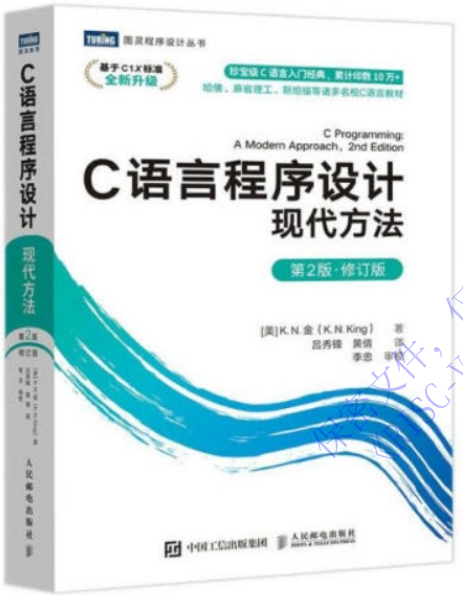
第4季

C语言陷阱精讲

版权文件，仅供奔跑吧Linux社区第4季视频课程《RISC-V体系结构与编程实践》的学员使用！
版权归奔跑吧Linux社区所有！
微信公众号：奔跑吧Linux社区

本节课主要内容

- 本章主要内容
  - C语言常见陷阱



本节课主要讲解书上第17章内容

版权文件，仅供购买了奔跑吧Linux社区第4季视频课程《RISC-V体系结构编程与实践》的学员浏览之用！
版权归奔跑吧Linux社区所有！
微信公众号：奔跑吧Linux社区

Part 1： C语言常见陷阱

保密文件，仅供购买了奔跑吧Linux社区第4季视频课程
《RISC-V体系结构编程与实战》的学员浏览之用！
版权归奔跑吧Linux社区所有
微信公众号：奔跑吧Linux社区

数据模型

- 32位处理器通常采用ILP32数据模型，而64位处理器可以采用LP64和ILP64数据模型。
- Linux系统采用LP64数据模型。

表 17.1　ILP32、ILP64、LP64 数据模型中不同数据类型的长度

数据类型	ILP32 数据模型中的长度/位	ILP64 数据模型中的长度/位	LP64 数据模型中的长度/位
char	8	8	8
short	16	16	16
int	32	64	32
long	32	64	64
long long	64	64	64
pointer	32	64	64
size_t	32	64	64
float	32	32	32
double	64	64	64

保密文件，仅供购买了奔跑吧Linux社区第4季视频课程
《Linux体系结构编程与实践》的用户浏览之用。
版权归奔跑吧Linux社区所有
微信公众号：奔跑吧Linux社区

强制类型转换导致问题

- 使用int或者unsigned int进行强制类型转换，在64位系统中可能会有问题。

例17-1

```
1  char * get_pte(char *pte_base, int offset)
2  {
3      int pte_addr, pte;
4
5      pte_addr = (int)pte_base;
6      pte = pte_addr + offset;
7
8      return (char *)pte;
9  }
```

使用int类型把pte_base指针转换成地址，在32位系统中这没有问题，但在64位处理器里就有问题

```
pte_addr = (intptr_t)pte_base;
```

推荐使用C99标准定义intptr_t和uintptr_t类型

```
#if __WORDSIZE == 64
    typedef long int            intptr_t;
    typedef unsigned long int   uintptr_t;
#else
    typedef int                 intptr_t;
    typedef unsigned int        uintptr_t;
#endif
```

保密文件，仅供购买了奔跑吧Linux社区第4季视频课程和《RISC-V体系结构编程与实践》的学员浏览之用！
版权归奔跑吧Linux社区所有！
微信公众号：奔跑吧Linux社区

Linux内核使用unsigned long

```
1   unsigned long __get_free_pages(gfp_t gfp_mask, unsigned int order)
2   {
3       struct page *page;
4
5       page = alloc_pages(gfp_mask & ~__GFP_HIGHMEM, order);
6       if (!page)
7           return 0;
8       return (unsigned long) page_address(page);
9   }
```

使用unsigned long转换内存地址，这基于指针和长整型的长度相等的事实

保密文件，仅供购买了奔跑吧Linux社区第一季视频课程《RISC-V体系结构编程与实践》的学员浏览使用！
版权归奔跑吧Linux社区所有
微信公众号：奔跑吧Linux社区

数据类型转换

- 在赋值表达式中，右边表达式的值自动隐式转换为左边变量的类型。

  a = b

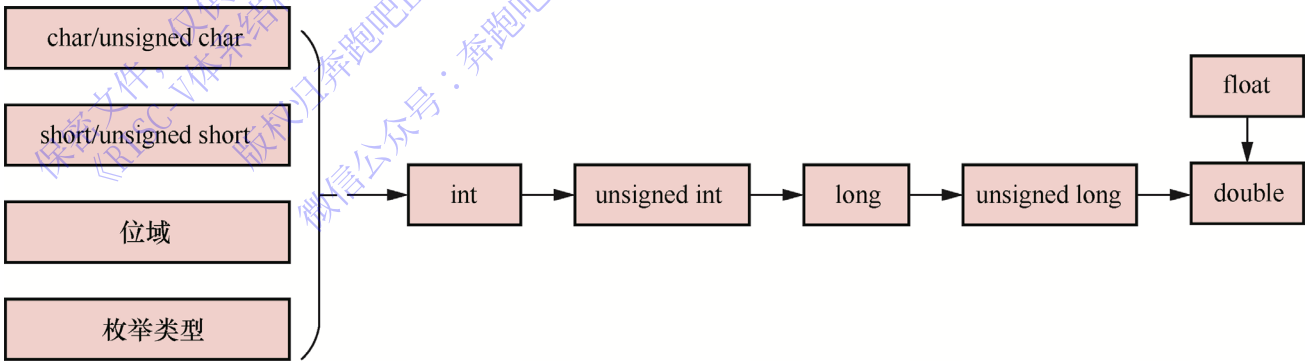
- 在算术表达式中，占字节少的数据类型向占字节多的数据类型转换

  (int)a + （long） b

- 在算术表达式中，当对有符号数据类型与无符号数据类型进行运算时，需要把有符号数据类型转换为无符号数据类型。

  (int)a + (unsigned int) b

- 整数常量通常属于int类型



保密文件，仅供购买了奔跑吧Linux社区第4期视频课程《基于RISC-V体系结构编程与实践》的学员浏览之用！
版权归奔跑吧Linux社区所有！
微信公众号：奔跑吧Linux社区

```
1  #include <stdio.h>
2
3  void main()
4  {
5     unsigned int i = 3;
6
7     printf("0x%x\n", i * -1);
8  }
```

-1是整数常量，它可以用int类型表达，而变量i属于unsigned int类型。

根据规则，当对int类型和unsigned int类型数据进行计算时，需要把int类型转换成unsigned int类型。所以，数据-1转换成unsigned int类型就是0xFFFF FFFF。于是，表达式“i * -1”变成“3 * 0xFFFF FFFF”，计算结果会溢出，最后变成0xFFFF FFFD。

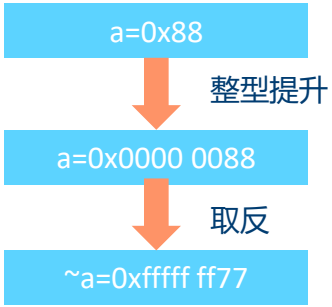
本资料为《奔跑吧Linux社区第4季视频课程》的学员专属资料，仅供购买了奔跑吧Linux社区第4季视频课程的学员浏览之用！
版权所有：奔跑吧Linux社区
微信公众号：奔跑吧Linux社区

整型提升（integral promotion）

- 在表达式中，当使用有符号或者无符号的char、short、位域（bit-field）以及枚举类型时，它们都应该提升到int类型。

```
1   #include <stdio.h>
2
3   void main()
4   {
5       char a;
6       unsigned int b;
7       unsigned long c;
8
9       a = 0x88;
10       b = ~a;
11       c = ~a;
12
13       printf("a=0x%x, ~a=0x%x, b=0x%x, c=0x%lx\n", a, ~a, b, c);
14  }
```

a=0x88, ~a=0xffffff77, b=0xffffff77, c=0xffffffffffffff77



保密文件，仅供购买了奔跑吧Linux社区第4季视频课程《RISC-V体系结构编程与实践》的学员浏览之用！
版权归奔跑吧Linux社区所有！
微信公众号：奔跑吧Linux社区

符号扩展

- 当要把一个带符号的整数提升为同一类型或更长类型的无符号整数时，它首先被提升为更长类型的带符号等价数值，然后转换为无符号值。

```
1   #include <stdio.h>
2
3   struct foo {
4      unsigned int a:19;
5      unsigned int b:13;
6   };
7
8   void main()
9   {
10     struct foo addr;
11
12     unsigned long base;
13
14     addr.a = 0x40000;
15     base = addr.a <<13;
16
17     printf("0x%x, 0x%lx\n", addr.a <<13, base);
18  }
```

0x80000000, 0xffffffff80000000

- addr.a是位域，要先转成int类型
- addr.a << 13 等于0x80000000，它是int类型
- addr.a << 13 要赋值给base，相当于int 类型赋值给 unsigned long。

有符号int类型 转换在unsigned long，要先转成long，然后在转成unsigned long

保密文件，仅供购买了奔跑吧Linux社区第4季视频课程《linux C编程体系结构编程与实践》的学员浏览之用！
版权归奔跑吧Linux社区所有！
微信公众号：奔跑吧Linux社区

```
1   #include <stdio.h>
2
3   struct foo {
4      unsigned int a:19;
5      unsigned int b:13;
6   };
7
8   void main()
9   {
10      struct foo addr;
11
12      unsigned long base;
13
14      addr.a = 0x40000;
15      base = (unsigned long)addr.a <<13;
16
17      printf("0x%x, 0x%lx\n", addr.a <<13, base);
18   }
```

想让base得到正确的值，可以先把addr.a从int类型转换成unsigned long类型。

保密文件，仅供购买了奔跑吧Linux社区第4季视频课程《基于RISC-V体系结构编程与实践》的学员浏览之用！
版权归奔跑吧Linux社区所有！
微信公众号：奔跑吧Linux社区

书上例17-5

```
#include <stdio.h>

void main()
{
    unsigned char a = 0xa5;
    unsigned char b = ~a>>4 + 1;

    printf("b=%d\n", b);
}
```

1. 在表达式“~a>>4 + 1”中，按位取反的优先级最高，首先计算“~a”表达式
2. 根据整型提升的规则，a被提升为int类型，最终得到0xFFFF FF5A
3. 加法的优先级高于右移运算的优先级，表达式变成0xFFFF FF5A >> 5，得到0xFFFF FFFA
4. 最终b的值为0xFA，即250。

保密文件（仅供购买了奔跑吧Linux社区第4季视频课程《RISC-V架构与编程与实践》的学员浏览之用！
版权归奔跑吧Linux社区所有！
微信公众号：奔跑吧Linux社区

移位操作

- 整数常量通常被看成int类型。如果移位的范围超过int类型，那么就会出错了。

```
#include <stdio.h>

void main()
{
    unsigned long reg = 1 << 33;

    printf("0x%lx\n", reg);
}
```

```
benshushu:mnt# gcc test.c -o test
test.c: In function ‘main’:
test.c:5:24: warning: left shift count >= width of type [-Wshift-count-overflow]
   5 |  unsigned long reg = 1 << 33;
     |
```

正确的做法是使用“1UL”，这样编译器会将这个整数常量的类型看成unsigned long类型

```
unsigned long reg = 1UL << 33;
```

保密文件，仅供购买了奔跑吧Linux社区第4季视频课程《RISC-V体系结构编程与实践》的学员浏览之用！
版权归奔跑吧Linux社区所有
微信公众号：奔跑吧Linux社区