More detailed

C 程序设计

C Programming

15

位处理

理论课程

厦門大學

XIAMEN UNIVERSITY

信息学院

(特色化示范性软件学院)

School of Informatics

黄 炜

博士、副教授

Wei Huang

知识框架

- 位操作符
- 位字段和联合

内容纲要

1	位操作符
2	位字段和联合

二进制数

- 人表示数：十根手指，十进制
  - $2157=2*10^3+1*10^2+5*10^1+7*10^0$
- 计算机表示数：高低电平，二进制
  - $100001101101=1*2^{11}+1*2^6+1*2^5+1*2^3+1*2^2+1*2^0$
- 值和进制的关系
- 八进制（Octal）
- 十六进制（Hexadecimal）
  - 0～9 仍对应 0～9
  - A～F 对应 10～15

位和字节

- 网络中传送信息通常以位为基本单位
- 内存中存储信息通常以字节为基本单位

C的位运算符

- 位逻辑运算符
  - 在二进制层面按位操作
  - 自与（&=）、自反（|=）、自异或（^=）

位操作	符号	对应算术	规则	代码示例
按位取反	~	相反 $1-x$	~0=1　~1=0	`newval = ~val;`
按位与	&	乘法	0&0=0　0&1=0 1&0=0　1&1=1	`val = val & 0377;`
按位或	|	加法	0|0=0　0|1=1 1|0=1　1|1=1	`val = val | 0377;`
按位异或	^	减法	0^0=0　0^1=1 1^0=1　1^1=0	`val = val ^ 0377;`

C的位运算符

- 位左右移运算符

位操作	符号	对应算术	规则	代码示例
左移	<<	乘以2次幂	1<<1=2 1<<2=4	onkoo = stonk << 2;
右移	>>	整除2次幂	8>>2=2 2>>1=1	onkoo = stonk >> 2;

  - 右移特殊性
    - 对于无符号整数，新位置补0。
    - 对于有符号整数，一部分系统补0，另一部分系统补“最高位”。

C的位运算符

- 掩码操作
  - 对目标字段进行位操作的一串二进制代码。
  - 打开位：将字节与掩码按位或
    - 示例：(10000000) | (00000101) ➔ (10000101)
    - 用法：

```
flags | MASK
```

  - 关闭位：将字节与掩码的反码进行按位与
    - 示例：(00001111) &~ (10110110) ➔ (00001001)
    - 用法

```
flags &= ~MASK;
```

C的位运算符

- 位逻辑运算符
  - 转置位：将字节与掩码按异或
    - 示例：(00001111) ^ (10110110) ➔ (10111001)
    - 用法

```
flags ^= MASK;
ch ^= 0xff; /* or ch ^= 0377; */
```

  - 获取位：&
    - 示例：(10010011) & (00000010) ➔ (00000010)
    - 用法：

```
if ((flags & MASK) == MASK)
```

```c
/* binbit.c -- using bit operations to display binary */
#include <stdio.h>
#include <limits.h>  // for CHAR_BIT, # of bits per char
char * itobs(int, char *);
void show_bstr(const char *);
int main(void)
{
    char bin_str[CHAR_BIT * sizeof(int) + 1];
    int number;
    puts("Enter integers and see them in binary.");
    puts("Non-numeric input terminates program.");
    while (scanf("%d", &number) == 1)
    {
        itobs(number,bin_str);
        printf("%d is ", number);
        show_bstr(bin_str);
        putchar('\n');
    }
    puts("Bye!");
    return 0;
}

char * itobs(int n, char * ps)
{
    int i;
    const static int size = CHAR_BIT * sizeof(int);
    for (i = size - 1; i >= 0; i--, n >>= 1)
        ps[i] = (01 & n) + '0';
    ps[size] = '\0';
    return ps;
}

/* show binary string in blocks of 4 */
void show_bstr(const char * str)
{
    int i = 0;
    while (str[i])  /* not the null character */
    {
        putchar(str[i]);
        if(++i % 4 == 0 && str[i])
            putchar(' ');
    }
}
```

```
Non-numeric input terminates program.
32↲
32 is 0000 0000 0000 0000 0000 0000 0010 0000
95↲
95 is 0000 0000 0000 0000 0000 0000 0101 1111
168654341↲
168654341 is 0000 1010 0000 1101 0111 0110 0000
0101
1↲
1 is 0000 0000 0000 0000 0000 0000 0000 0001
-1↲
-1 is 1111 1111 1111 1111 1111 1111 1111 1111
0↲
0 is 0000 0000 0000 0000 0000 0000 0000 0000
q↲
Bye!
```

```c
/* invert4.c -- using bit operations to display binary */
#include <stdio.h>
#include <limits.h>
char * itobs(int, char *);
void show_bstr(const char *);
int invert_end(int num, int bits);

int main(void)
{
    char bin_str[CHAR_BIT * sizeof(int) + 1];

    int number;

    puts("Enter integers and see them in binary.");
    puts("Non-numeric input terminates program.");
    while (scanf("%d", &number) == 1)
    {
        itobs(number,bin_str);
        printf("%d is\n", number);
        show_bstr(bin_str);
        putchar('\n');
        number = invert_end(number, 4);
        printf("Inverting the last 4 bits gives\n");
        show_bstr(itobs(number,bin_str));
        putchar('\n');
    }
    puts("Bye!");

    return 0;
}

char * itobs(int n, char * ps)
{
    int i;
    const static int size = CHAR_BIT * sizeof(int);
    for (i = size - 1; i >= 0; i--, n >>= 1)
        ps[i] = (01 & n) + '0';
    ps[size] = '\0';

    return ps;
}
/* show binary string in blocks of 4 */
void show_bstr(const char * str)
{
    int i = 0;

    while (str[i])  /* not the null character */
    {
        putchar(str[i]);
        if(++i % 4 == 0 && str[i])
            putchar(' ');
    }
}
```

```c
int invert_end(int num, int bits)
{
    int mask = 0;
    int bitval = 1;

    while (bits-- > 0)
    {
        mask |= bitval;
        bitval <<= 1;
    }

    return num ^ mask;
}
```

```
Enter integers and see them in binary.
Non-numeric input terminates program.
95↲
95 is
0000 0000 0000 0000 0000 0000 0101 1111
Inverting the last 4 bits gives
0000 0000 0000 0000 0000 0000 0101 0000
168654341↲
168654341 is
0000 1010 0000 1101 0111 0110 0000 0101
Inverting the last 4 bits gives
0000 1010 0000 1101 0111 0110 0000 1010
1↲
1 is
0000 0000 0000 0000 0000 0000 0000 0001
Inverting the last 4 bits gives
0000 0000 0000 0000 0000 0000 0000 1110
q↲
Bye!
```

运算符的优先级

序号	符号	说明
1 →	后缀++ --	后缀增减量
	( )	函数调用
	[ ]	数组下标
	.	结构体联合体成员
	->	结构/联合指针成员
	(type){list}	复合文字
2 ←	前缀++ --	前缀增减量
	+ -	正负号
	!	逻辑运算符：非
	~	位运算符：非
	(type)	强制类型转换
	*	间接寻址
	&	取址
	sizeof	存储空间
3	* / %	算术运算：乘除模

序号	符号	说明
4	+ -	算术运算：加减
5	<< >>	位运算符：左右移
6	< > <= >=	关系运算符：大小
7	== !=	关系运算符：相等
8	&	位运算符：与
9	^	位运算符：异或
10	|	位运算符：或
11	&&	逻辑运算符：与
12	||	逻辑运算符：或
13	?:	三元条件运算符
14 ←	=	赋值
	+= -= *= /= %=	自增、自减、自乘、自除、自模
	<<= >>= &= ^= |=	自左右移、自与、自异或、自或
15	,	逗号表达式

运算符的优先级

- 一元＜二元（除赋值和逗号）＜三元＜赋值＜逗号
  - 一元（部分）、三元、赋值为自右向左结合
    - 前缀、正负、非、类型转换、指针、空间
  - 一元（部分）、二元（除赋值外）为自左向右
    - 后缀、指向（函数、数组元素、成员）、复合文字
- 二元：算术＜移位＜关系＜位（除移位外）＜逻辑
  - 算术：乘除模＜加减
  - 关系：大小＜相等
  - 位：与（相当于乘）＜异或（相当于减）＜或（相当于加）
  - 逻辑：与（相当于乘）＜或（相当于加）

内容纲要

1 位操作符

2 位字段和联合

位字段

- 位字段是有符号或无符号整型中一组相邻的位

```
struct box_props {
    bool opaque                  : 1;   // or unsigned int (pre C99)
    unsigned int fill_color      : 3;
    unsigned int                 : 4;
    bool show_border             : 1;   // or unsigned int (pre C99)
    unsigned int border_color    : 3;
    unsigned int border_style    : 2;
    unsigned int                 : 2;
};
```

- 在内存里如何存储？
  - 自低向高、不跨边界、尽量无缝连接、允许省略变量名

```
/* fields.c -- define and use fields */
#include <stdio.h>
#include <stdbool.h>   //C99, defines bool, true, false

/* line styles     */
#define SOLID   0
#define DOTTED  1
#define DASHED  2
/* primary colors  */
#define BLUE    4
#define GREEN   2
#define RED     1
/* mixed colors    */
#define BLACK   0
#define YELLOW  (RED | GREEN)
#define MAGENTA (RED | BLUE)
#define CYAN    (GREEN | BLUE)
#define WHITE   (RED | GREEN | BLUE)
```

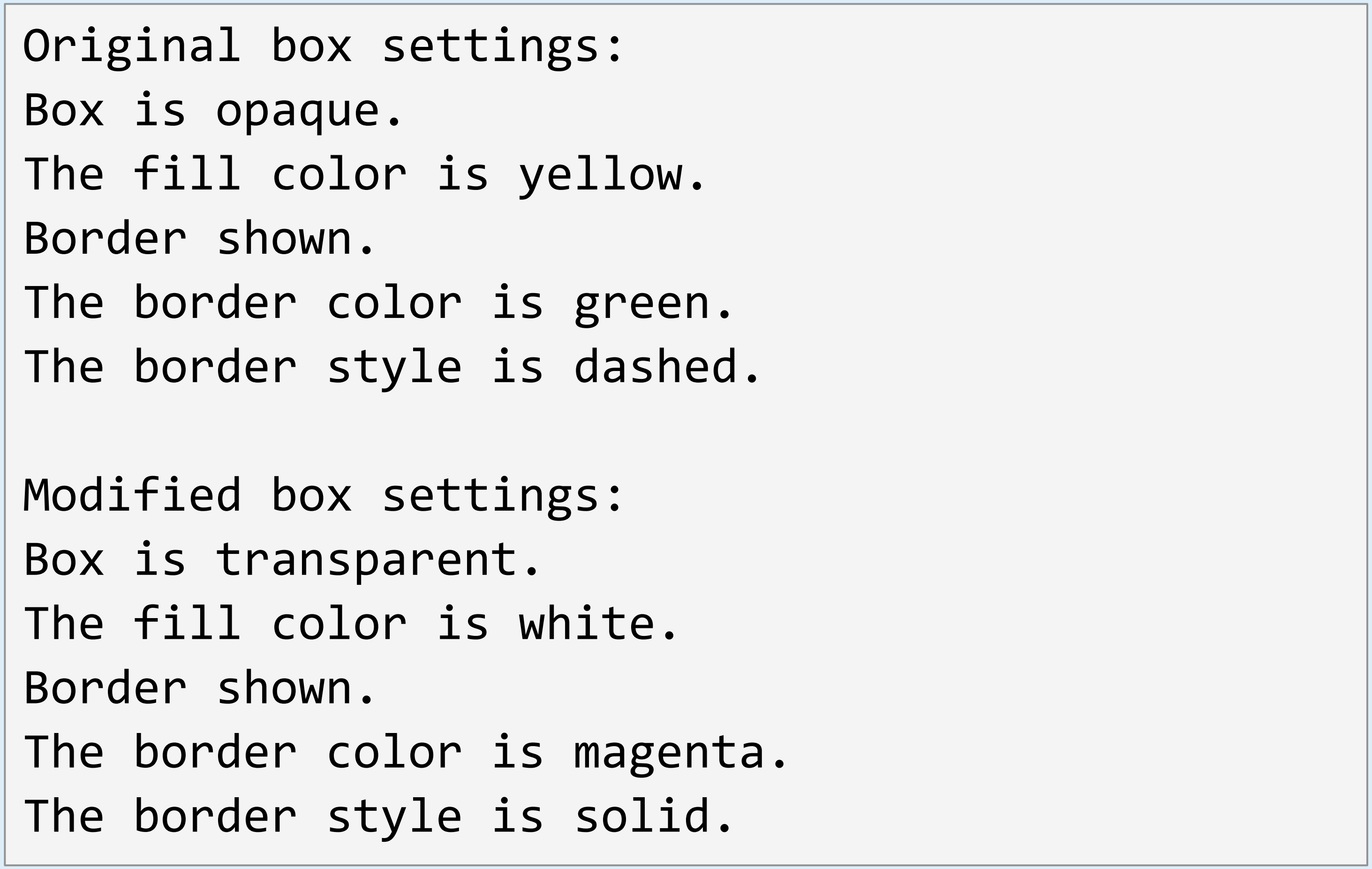
```c
const char * colors[8] = {"black", "red", "green", "yellow",
    "blue", "magenta", "cyan", "white"};
struct box_props {
    bool opaque                  : 1;   // or unsigned int (pre C99)
    unsigned int fill_color      : 3;
    unsigned int                 : 4;
    bool show_border             : 1;   // or unsigned int (pre C99)
    unsigned int border_color    : 3;
    unsigned int border_style    : 2;
    unsigned int                 : 2;
};

void show_settings(const struct box_props * pb);

int main(void)
{
    /* create and initialize box_props structure */
    struct box_props box = {true, YELLOW , true, GREEN,
DASHED};
    printf("Original box settings:\n");
    show_settings(&box);
    box.opaque = false;
    box.fill_color = WHITE;
    box.border_color = MAGENTA;
    box.border_style = SOLID;
    printf("\nModified box settings:\n");
    show_settings(&box);

    return 0;
}

void show_settings(const struct box_props * pb)
{
    printf("Box is %s.\n", pb->opaque == true ? "opaque":
"transparent");
    printf("The fill color is %s.\n", colors[pb->fill_color]);
    printf("Border %s.\n", pb->show_border == true ? "shown" :
"not shown");
    printf("The border color is %s.\n", colors[pb-
>border_color]);
    printf ("The border style is ");
    switch(pb->border_style)
    {
        case SOLID  : printf("solid.\n"); break;
        case DOTTED : printf("dotted.\n"); break;
        case DASHED : printf("dashed.\n"); break;
        default     : printf("unknown type.\n");
    }
}
```



```
Original box settings:
Box is opaque.
The fill color is yellow.
Border shown.
The border color is green.
The border style is dashed.

Modified box settings:
Box is transparent.
The fill color is white.
Border shown.
The border color is magenta.
The border style is solid.
```

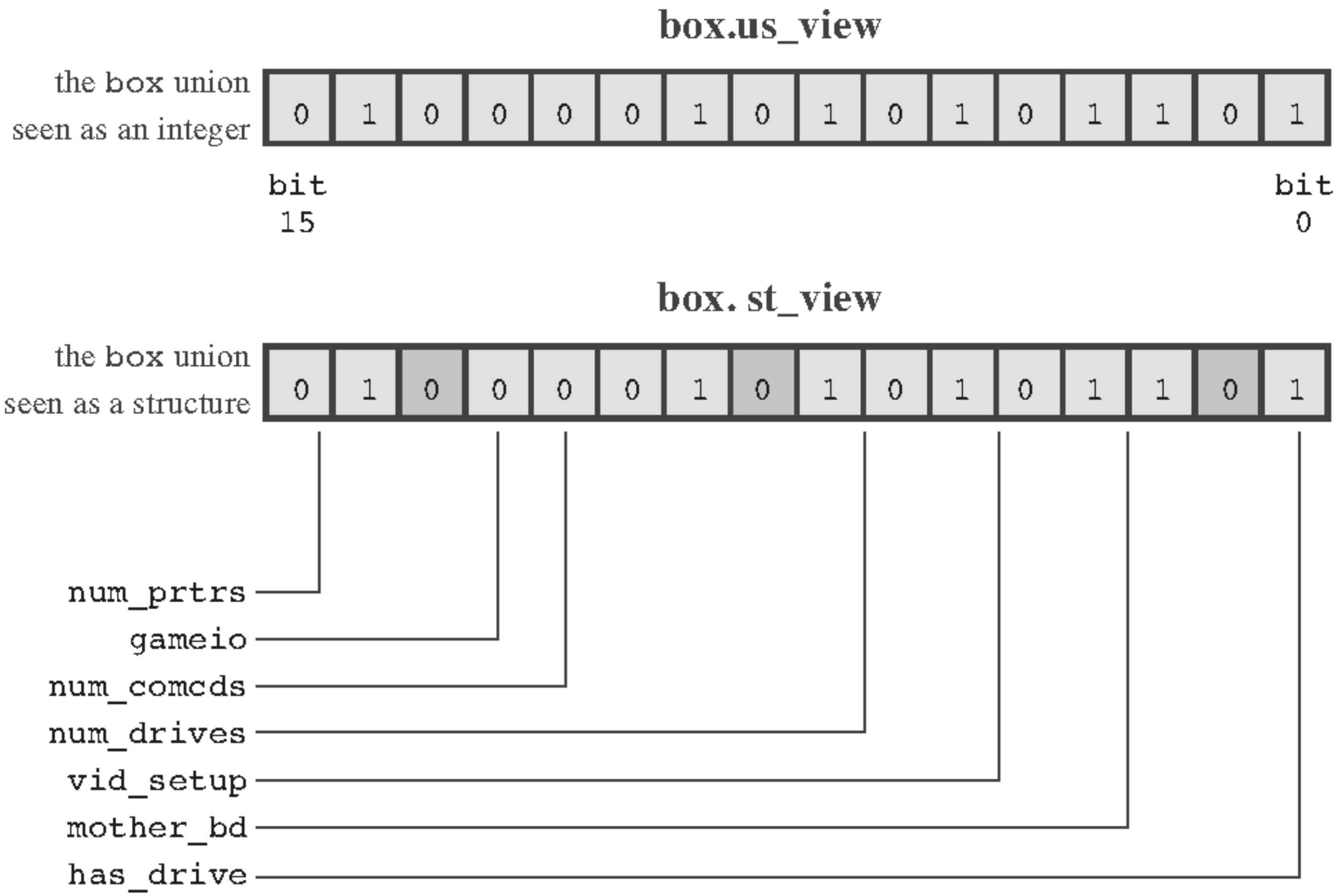
位字段与联合

- 见下方实例

```
struct box_props {
    bool opaque                 : 1;
    unsigned int fill_color     : 3;
    unsigned int                : 4;
    bool show_border            : 1;
    unsigned int border_color   : 3;
    unsigned int border_style   : 2;
    unsigned int                : 2;
};
union Views     /* look at data as struct or as unsigned short */
{
    struct box_props st_view;
    unsigned short us_view;
};
```

位字段与联合

- 例题：整数与结构的联合



box.us_view

the box union seen as an integer

0 1 0 0 0 0 1 0 1 0 1 0 1 1 0 1

bit 15

bit 0

box. st_view

the box union seen as a structure

0 1 0 0 0 0 1 0 1 0 1 0 1 1 0 1

num_prtrs

gameio

num_comcds

num_drives

vid_setup

mother_bd

has_drive

```c
/* dualview.c -- bit fields and bitwise operators */
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
/* BIT-FIELD CONSTANTS */
/* line styles     */
#define SOLID   0
#define DOTTED  1
#define DASHED  2
/* primary colors  */
#define BLUE    4
#define GREEN   2
#define RED     1
/* mixed colors    */
#define BLACK   0
#define YELLOW  (RED | GREEN)
#define MAGENTA (RED | BLUE)
#define CYAN    (GREEN | BLUE)
#define WHITE   (RED | GREEN | BLUE)
```

```
/* BITWISE CONSTANTS   */
#define OPAQUE          0x1
#define FILL_BLUE       0x8
#define FILL_GREEN      0x4
#define FILL_RED        0x2
#define FILL_MASK       0xE
#define BORDER          0x100
#define BORDER_BLUE     0x800
#define BORDER_GREEN    0x400
#define BORDER_RED      0x200
#define BORDER_MASK     0xE00
#define B_SOLID         0
#define B_DOTTED        0x1000
#define B_DASHED        0x2000
#define STYLE_MASK      0x3000

const char * colors[8] = {"black", "red", "green", "yellow",
    "blue", "magenta", "cyan", "white"};
```

```
struct box_props {

    bool opaque                 : 1;
    unsigned int fill_color     : 3;
    unsigned int                : 4;
    bool show_border            : 1;
    unsigned int border_color   : 3;
    unsigned int border_style   : 2;
    unsigned int                : 2;
};
union Views     /* look at data as struct or as unsigned short */
{
    struct box_props st_view;
    unsigned short us_view;
};

void show_settings(const struct box_props * pb);
void show_settings1(unsigned short);
char * itobs(int n, char * ps);
```

```c
int main(void)
{
    /* create Views object, initialize struct box view */
    union Views box = {{true, YELLOW , true, GREEN, DASHED}};
    char bin_str[8 * sizeof(unsigned int) + 1];

    printf("Original box settings:\n");
    show_settings(&box.st_view);
    printf("\nBox settings using unsigned int view:\n");
    show_settings1(box.us_view);

    printf("bits are %s\n",
           itobs(box.us_view,bin_str));
    box.us_view &= ~FILL_MASK;          /* clear fill bits */
    box.us_view |= (FILL_BLUE | FILL_GREEN); /* reset fill */
    box.us_view ^= OPAQUE;               /* toggle opacity */
    box.us_view |= BORDER_RED;           /* wrong approach */
    box.us_view &= ~STYLE_MASK;        /* clear style bits */
    box.us_view |= B_DOTTED;        /* set style to dotted */
    printf("\nModified box settings:\n");
    show_settings(&box.st_view);
    printf("\nBox settings using unsigned int view:\n");
    show_settings1(box.us_view);
    printf("bits are %s\n",itobs(box.us_view,bin_str));
    return 0;
}

void show_settings(const struct box_props * pb)
{
    printf("Box is %s.\n",pb->opaque == true ? "opaque":
"transparent");
    printf("The fill color is %s.\n", colors[pb->fill_color]);
    printf("Border %s.\n",pb->show_border == true ? "shown" :
"not shown");
    printf("The border color is %s.\n", colors[pb-
>border_color]);
    printf ("The border style is ");
```

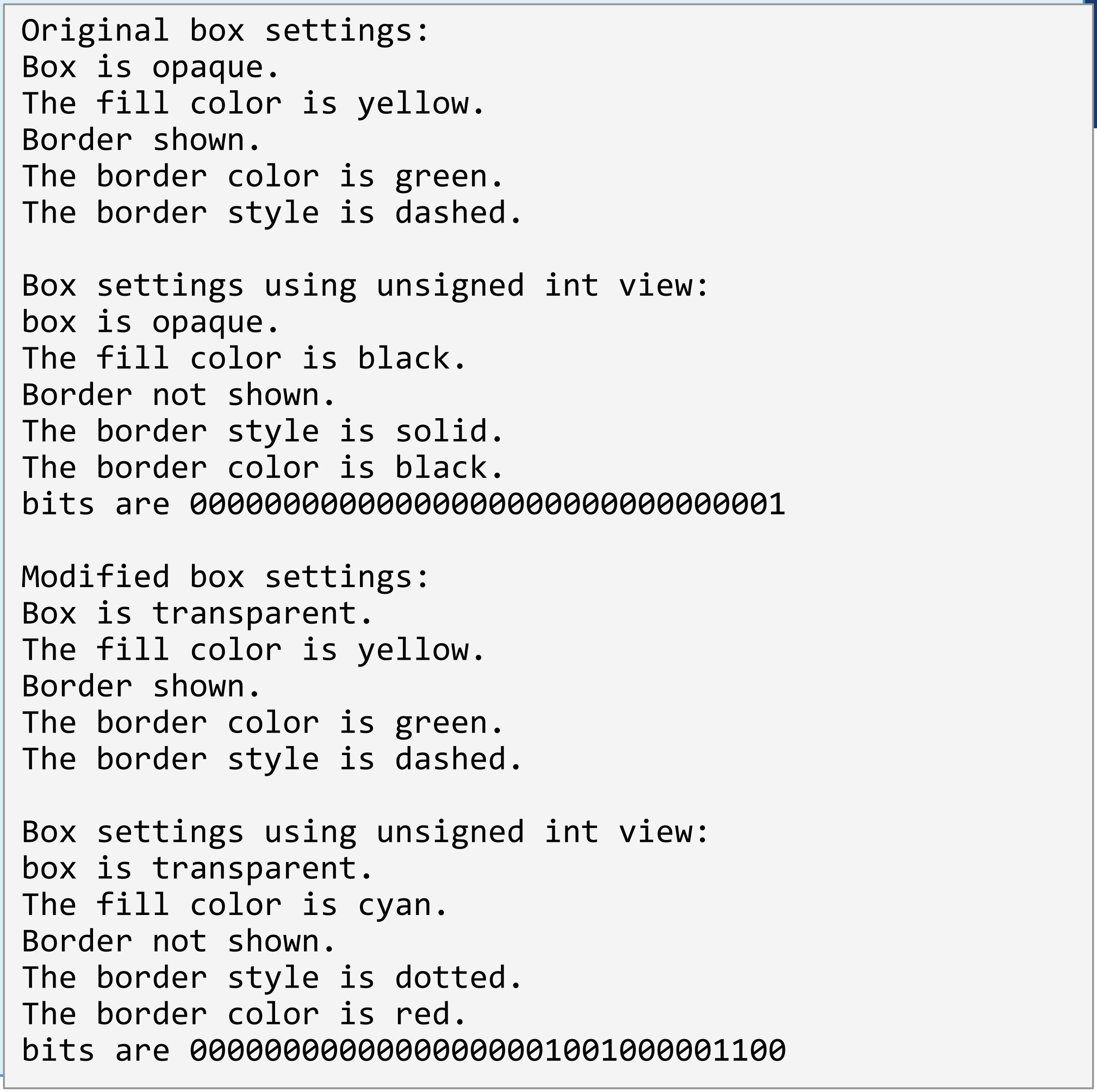
```c
    switch(pb->border_style)
    {
        case SOLID  : printf("solid.\n"); break;
        case DOTTED : printf("dotted.\n"); break;
        case DASHED : printf("dashed.\n"); break;
        default     : printf("unknown type.\n");
    }
}

void show_settings1(unsigned short us)
{
    printf("box is %s.\n",
           (us & OPAQUE) == OPAQUE? "opaque": "transparent");
    printf("The fill color is %s.\n",
           colors[(us >> 1) & 07]);
    printf("Border %s.\n",
           (us & BORDER) == BORDER? "shown" : "not shown");
    printf ("The border style is ");
    switch(us & STYLE_MASK)
    {
        case B_SOLID  : printf("solid.\n"); break;
        case B_DOTTED : printf("dotted.\n"); break;
        case B_DASHED : printf("dashed.\n"); break;
        default       : printf("unknown type.\n");
    }
    printf("The border color is %s.\n",
           colors[(us >> 9) & 07]);
}

char * itobs(int n, char * ps)
{
    int i;
    const static int size = CHAR_BIT * sizeof(int);

    for (i = size - 1; i >= 0; i--, n >>= 1)
        ps[i] = (01 & n) + '0';
    ps[size] = '\0';

    return ps;
}
```



```
Original box settings:
Box is opaque.
The fill color is yellow.
Border shown.
The border color is green.
The border style is dashed.

Box settings using unsigned int view:
box is opaque.
The fill color is black.
Border not shown.
The border style is solid.
The border color is black.
bits are 00000000000000000000000000000001

Modified box settings:
Box is transparent.
The fill color is yellow.
Border shown.
The border color is green.
The border style is dashed.

Box settings using unsigned int view:
box is transparent.
The fill color is cyan.
Border not shown.
The border style is dotted.
The border color is red.
bits are 00000000000000000001001000001100
```

位字段与联合

- 位字段和按位视图的区别在于后者需要记住位置信息
- 位字段和位的位置之间的对应关系是依赖于实现的。
- 大端（尾）序和小端序
  - 如：对于long型数据0x12345678，按书写习惯0x12最大，0x78最小，
  - 大端序：最大的存在尾部（内存地址较低处），小端反之
    - 3000H：0x12；3001H：0x34；3002H：0x56；3003H：0x78

```c
//  align.c -- using _Alignof and _Alignas  (C11)
#include <stdio.h>
int main(void) {
    double dx;
    char ca;
    char cx;
    double dz;
    char cb;
    char _Alignas(double)  cz;
    printf("char alignment:   %zd\n", _Alignof(char));
    printf("double alignment: %zd\n", _Alignof(double));
    printf("&dx: %p\n", &dx);
    printf("&ca: %p\n", &ca);
    printf("&cx: %p\n", &cx);
    printf("&dz: %p\n", &dz);
    printf("&cb: %p\n", &cb);
    printf("&cz: %p\n", &cz);
    return 0;
}
```

```
char alignment:   1
double alignment: 8
&dx: 0028FEE8
&ca: 0028FEE7
&cx: 0028FEE6
&dz: 0028FED8
&cb: 0028FED7
&cz: 0028FED0
```

C程序设计
C Programming

15

谢谢观看

理论课程

厦門大學
XIAMEN UNIVERSITY

信息学院
(特色化示范性软件学院)
School of Informatics

黄炜
博士，副教授
Wei Huang