# 大端模式和小端模式笔记

## 定义

* 小端模式(little endian)：低字节放在低地址

例如：

int i =1;

在内存中表示为：

二进制：0000 0001 0000 0000 0000 0000 0000 0000

十六进制：01 00 00 00

* 大端模式(big endian)：高字节放在低地址

例如：

int i =1;

在内存中表示为：

二进制：0000 0000 0000 0000 0000 0000 0000 0001

十六进制：00 00 00 01

## 判断方法

### 判断整型的第一个字节是否为低位

#### 例程1

const int endian = 1;

#define is\_littleendian (1 == \*(char\*)&endian)

#### 例程2

int i = 0x11223344;

char\* p = (char\*)&i;

if(0x44 == \*p)

{

cout<<"little endian"<<endl;

}

else

{

cout<<"big endian"<<endl;

}

#### 例程3：利用union共享内存的特性

union

{

char str;

int data;

};

data = 0x11223344;

if(0x44 == str)

{

cout<<"little endian"<<endl;

}

else

{

cout<<"big endian"<<endl;

}

#### 例程4：利用union共享内存的特性

static union { char c[4]; unsigned long mylong; } endian\_test = {{ 'l', '?', '?', 'b' } };

#define ENDIANNESS ((char)endian\_test.mylong)

## 对比

小端模式：强制转换数据不需要调整字节内容。

大端模式：符号位的判定固定为第一个字节，容易判断正负。

## 常见的字节序

一般操作系统都是小端，而java和通讯协议是大端的。

### 常见CPU的字节序

Big Endian : PowerPC、IBM、Sun

Little Endian : x86、DEC

ARM既可以工作在大端模式，也可以工作在小端模式。

### 常见文件的字节序

Adobe PS – Big Endian

BMP – Little Endian

DXF(AutoCAD) – Variable

GIF – Little Endian

JPEG – Big Endian

MacPaint – Big Endian

RTF – Little Endian

## 大小端转换

### 对于字数据

typedef unsigned short uint16;

#define BigtoLittle16(A) (( ((uint16)(A) & 0xff00) >> 8) | (( (uint16)(A) & 0x00ff) << 8))

unsigned short i = 0x1122;

printf("0x%04X\n", i);//0x1122

unsigned short j = BigtoLittle16(i);

printf("0x%04X\n", j);//0x2211

### 对于双字数据

typedef unsigned int uint32;

#define BigtoLittle32(A) ( (((uint32)(A) & 0xff000000) >> 24) | \

(( (uint32)(A) & 0x00ff0000) >> 8) | \

(( (uint32)(A) & 0x0000ff00) << 8) | \

(( (uint32)(A) & 0x000000ff) << 24) )

unsigned int i = 0x11223344;

printf("0x%08X\n", i);//0x11223344

unsigned int j = BigtoLittle32(i);

printf("0x%08X\n", j);//0x44332211