



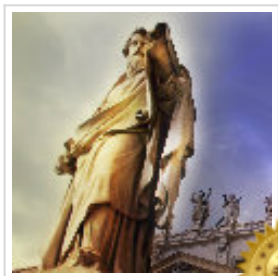
有奖征集：文集--博客系列博文管理

wjlkoorey的博客

wjlkoorey.blog.chinaunix.net

wanderlust in the sea...

首页 | 博文目录 | 关于我



wjlkoorey258



博客访问：509979

博文数量：97

博客积分：671

博客等级：上尉

原创

Linux环境下网络编程杂谈

2012-07-11 22:49:23

分类：LINUX

今天我们说说“Pre-网络编程”。内容比较杂，但都是在做网络应用程序开发过程中经常要遇到的问题。

一、大端、小端和网络字节序

小端字节序：little-endian，将低字节存放在内存的起始地址；

大端字节序：big-endian，将高字节存放在内存的其实地址。

例如，数字index=0x11223344，在大小端字节序方式下其存储形式为：

技术积分：10228

用户组：普通用户

注册时间：2010-12-18 16:08

加关注

短消息

论坛

加好友

个人简介

www.5678520.com

文章分类

全部博文 (97)

Netfilter&ebtabl (0)

算法设计 (8)

计算机系统 (11)

商海ABC (1)

存储 (6)

翻译 (3)

Java (1)

内核源码 (5)

其他 (3)

多媒体 (7)

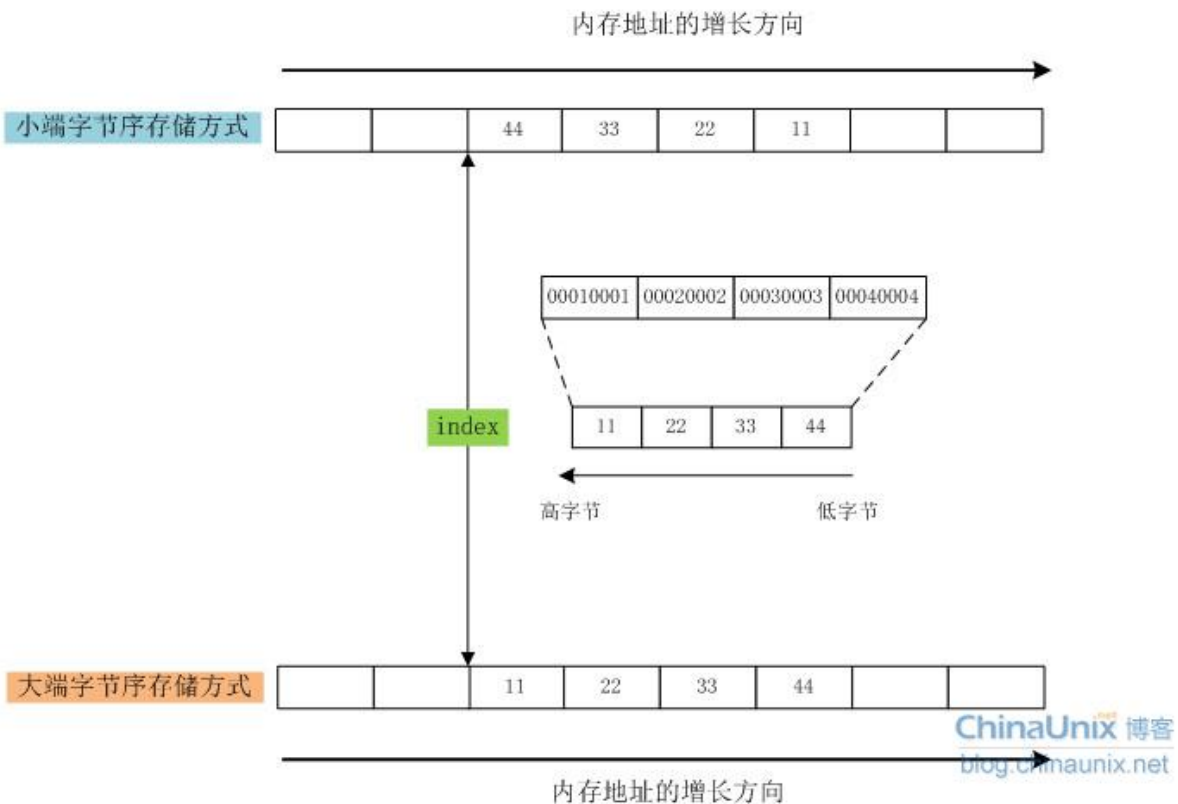
网络编程 (8)

系统管理 (3)

SNMP (2)

Netfilter和iptables (0)

未分配的博文 (39)



上图一目了然的可以看出大小端字节序的区别。

还有另外一个概念就是网络字节序。网络字节顺序是TCP/IP中规定好的一种数据表示格式，它与具体的CPU类型、操作系统等无关，从而可以保证数据在不同主机之间传输时能够被正确解释。网络字节顺序采用big endian方式。注意：X86系列CPU都是小端little-endian字节序，即低字节存低位，高字节存高位。

为此，Linux专门提供了字节转换函数。

```
unsigned long int htonl(unsigned long int hostlong)
unsigned short int htons(unsigned short int hostshort)
unsigned long int ntohl(unsigned long int netlong)
unsigned short int ntohs(unsigned short int netshort)
```

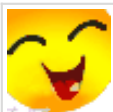
文章存档

2014年 (12)

2013年 (21)

2012年 (64)

我的朋友



2005227



lbird_11



tyj19891



lawrence



mzh2100



huangba



liucaipi

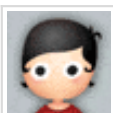


Zackary1

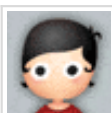


WbBullFr

最近访客



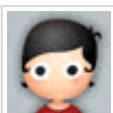
acorpe



雷锋不谢



y841618



cym0417



猪也有春



米娜拉夜

在这四个转换函数中，h代表host，n代表 network，s代表short，l代表long。htonl()函数的意义是将本机器上的long数据转化为网络上的long。其他几个函数的意义也差不多。

看个例子：

```
[koorey@localhost TCP]$ cat test.c
#include <stdio.h>
#include <arpa/inet.h>

int main(){
    long x = 0x11223344;
    long y ;
    char *c = &x;
    printf("H = %x %x %x %x\n",*c,*c+1,*c+2,*c+3));

    y = htonl(x);
    c = &y;
    printf("N = %x %x %x %x\n",*c,*c+1,*c+2,*c+3));
    return 0;
}
[koorey@localhost TCP]$ gcc -w -o test test.c
[koorey@localhost TCP]$ ./test
H = 44 33 22 11
N = 11 22 33 44
[koorey@localhost TCP]$
```

ChinaUnix 博客
blog.chinaunix.net

也就是说对于从网络上接收到的非单字节的的基本数据类型数据，首先需要用ntohl(s)将其转换成本地字节序；同理，发往网络的非单字节的的基本数据类型数据，首先用htonl(s)将其转换成网络字节序。这里最常见的就是IP地址和端口号。

二、点分十进制格式的IP地址和32bit的IP地址

我们常见的IP地址都是以点分十进制格式表示，例如“172.18.1.231”。而在程序中基本是



hmily36



VEGETA



phoenixc

以如下的结构表示一个IP：

```
struct in_addr {
    __be32    s_addr; //其实就是一个32bit的数字
};
```

它和点分十进制格式的IP地址可以通过一组API实现相互转换：

```
int inet_aton(const char *cp,struct in_addr *inp) 无效的地址cp则返回0；否则返回非0
char *inet_ntoa(struct in_addr in) 将一个32位的IP地址转换成点分十进制字符串。
```

这两个函数所要求的struct in_addr{}参数均为网络字节序。

继续看例子：

订阅



推荐博文

·模仿之中也少不了创新——Leo...

·学习Swift之(二)：swift开发...

·学习Swift之(一)：关于swift...

·实现dup2函数,要求不使用fcntl...

·LR模型的Spark实现

·对Oracle高水位线的研究实践...

·为学习Hadoop使用VMware准备3...

·【故障处理】opmn启动失败及...

·oracle 11g ASM 磁盘组在线扩...

·数据迁移中的数据库检查和建...

热词专题

·李体育老师荣获“杰出传承人...

·string.h

·安装oracle

·VMware VDP

·Android + 系统属性

```
[koorey@localhost TCP]$ cat test2.c
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char** argv){
    struct in_addr addr;
    int ret;
    char *c = NULL;

    if((ret = inet_aton(argv[1], &addr))){
        printf("addr = %x\n", addr.s_addr);
        c = (char*)&addr;
        printf("IP = %x.%x.%x.%x\n", *(c)&0xff, *(c+1)&0xff, *(c+2)&0xff, *(c+3)&0xff);
        printf("IP2 = %u.%u.%u.%u\n", *(c)&0xff, *(c+1)&0xff, *(c+2)&0xff, *(c+3)&0xff);
    }else{
        printf("Invalid!\n");
    }

    return 0;
}

[koorey@localhost TCP]$ gcc -w -o test2 test2.c
[koorey@localhost TCP]$ ./test2 "192.168.11.23"
addr = 170ba8c0
IP = c0.a8.b.17
IP2 = 192.168.11.23
[koorey@localhost TCP]$
```

ChinaUnix 博客
blog.chinaunix.net

“192.168.11.23”转换成数字就是0xc0a80b17，是网络字节序的。如果直接打印，那么本地按照小端字节序来输出，结果为net addr = 170ba8c0，刚好和实际相反。当我们先将其转换成本地字节序，然后再输出时结果就OK了，即host addr = c0a80b17。同理，inet_ntoa()也类似。

三、网络主机名和IP地址的对应关系

在做网络编程时经常要碰到的一个问题就是根据对方主机名来获取其IP地址，或者根据IP地址反过来解析主机名和其他信息。Linux提供了两个常用的API：

```
struct hostent *gethostbyname(const char *name);
```

```
struct hostent *gethostbyaddr(const void *addr, int len, int type);
```

这两个函数失败时返回NULL且设置h_errno错误变量，调用hstrerror(h_errno)或者herror("Error");可以得到详细的出错信息。成功时均返回如下结构：

```
struct hostent {
    char  *h_name;      /* 主机名*/
    char  **h_aliases;  /* 主机别名的列表*/
    int   h_addrtype;   /* 地址类型，AF_INET或其他*/
    int   h_length;     /* 所占的字节数，IPv4地址都是4字节 */
    char  **h_addr_list; /* IP地址列表，网络字节序*/
}

#define h_addr h_addr_list[0] /*后向兼容 */
```

gethostbyname可以将机器名(如www.google.com)转换为一个结构指针，gethostbyaddr可以将一个32位的IP地址(C0A80001)转换为结构指针。对于gethostbyaddr函数来说，输入参数“addr”的类型根据参数“type”来确定，目前type仅取AF_INET或AF_INET6。例如，type=2(即AF_INET)，则addr就必须为struct in_addr{}类型。

继续看例子：

```
#include <stdio.h>
#include <netdb.h>
#include <error.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
```



```
#include <arpa/inet.h>

int main(int arg,char** argv){
    struct hostent *host,*host2;
    if(NULL == (host = gethostbyname(argv[1]))){
        perror("Error");
        return 1;
    }

    printf("name = %s\n",host->h_name);
    printf("aliases = %s\n",*host->h_aliases);
    printf("add type = %d\n",host->h_addrtype);
    printf("len = %d\n",host->h_length);
    printf("IP=%s\n",inet_ntoa(*(struct in_addr*)host->h_addr));

    printf("=====\n");
    struct in_addr maddr;
    if(0 == inet_aton(argv[2],&maddr)){
        return 0;
    }

    char* c = (char*)&maddr;
```

```
printf("org =
%x.%x.%x.%x\n",*(c)&0xff,*(c+1)&0xff,*(c+2)&0xff,*(c+3)&0xff);

if(NULL == (host2 = gethostbyaddr(&maddr,4,2))){
    printf("Error:%s\n",hstrerror(h_errno));
    return 1;
}

printf("name = %s\n",host2->h_name);
printf("aliases = %s\n",*host2->h_aliases);
printf("add type = %d\n",host2->h_addrtype);
printf("len = %d\n",host2->h_length);
printf("IP=%s\n",inet_ntoa(*(struct in_addr*)host2->h_addr));

return 0;
}
```

运行结果如下：


```
[koorey@localhost TCP]$ ./test3 "www.google.com.hk" "181.23.46.131"
name = www-wide.l.google.com
aliases = www.google.com.hk
add type = 2
len = 4
IP=74.125.71.199
=====
org = b5.17.2e.83
name = 181-23-46-131.speedy.com.ar
aliases = (null)
add type = 2
len = 4
IP=181.23.46.131
[koorey@localhost TCP]$ ping www.chinaunix.net
PING www.chinaunix.net (61.55.167.120) 56(84) bytes of data.

--- www.chinaunix.net ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1038ms

[koorey@localhost TCP]$ ./test3 "www.google.com.hk" "61.55.167.120"
name = www-wide.l.google.com
aliases = www.google.com.hk
add type = 2
len = 4
IP=74.125.71.199
=====
org = 3d.37.a7.78
Error: Unknown host
```




当我们调用`gethostbyaddr`根据CU主页的IP地址获取其站点信息时返回的错误是“未知的主机”错误，原因留给大家自己思考吧。这充分说明对于`gethostbyname()`函数和`gethostbyaddr()`函数的调用一定要判断其返回值。

阅读(6756) | 评论(6) | 转发(30) |

上一篇：SNMP从入门到开发：进阶篇


下一篇：Linux网络编程：基于TCP的程序开发回顾篇

相关热门文章


 轻量级web server Tornado代码...


 linux 常见服务端口


 kernel 报错l701.exe[16922]:...

 unix网络编程一卷 unip.h...


 【ROOTFS搭建】busybox的httpd...


 C语言 如何在一个整型左边补0...

 杂谈icmp


 xmanager 2.0 for linux配置

 python无法爬取阿里巴巴的数据...

 电子商务网站的六大制作技巧...

 什么是shell

 linux-2.6.28 和linux-2.6.32....

 “万万没想到”惊现《我叫MT...

 linux socket的bug??

 linux su - username -c 命...

给主人留下些什么吧！~~



ruanben

2014-07-02 21:07:37

```
#if defined(__LITTLE_ENDIAN_BITFIELD)
__u16 res1:4,
doff:4,
fin:1,
syn:1,
rst:1,
psh:1,
ack:1,
urg:1,
ece:1,
cwr:1;
#elif defined(__BIG_ENDIAN_BITFIELD)
__u16 doff:4,
res1:4,
cwr:1,
ece:1,
urg:1,
ack:1,
psh:1,
rst:1,
syn:1,
```

```
fin:1;  
#else
```

[回复](#) | [举报](#)

xingfengshi 2014-04-13 21:14:25

fanxiao_cn : 大神，为什么“`printf("IP = %x.%x.%x.%x\n", *(c)&0xff, *(c+1)&0xff, *(c+2), *(c+3));`”里面*c和*(c+1)要&0xff,我试了不与的话像*c返回值会成为0xffffffffc0,为什么呢？第一个例子中的0x11223344 为什么就不需要与0xff?

同问啊~~~!!!~!·!

[回复](#) | [举报](#)

xingfengshi 2014-04-12 23:16:39

写得不错，对于一些问题的理解很有帮助哦

[回复](#) | [举报](#)

fanxiao_cn 2013-04-26 06:07:49

fanxiao_cn : 大神，为什么“`printf("IP = %x.%x.%x.%x\n", *(c)&0xff, *(c+1)&0xff, *(c+2), *(c+3));`”里面*c和*(c+1)要&0xff,我试了不与的话像*c返回值会成为0xffffffffc0,为什么呢？第一个例子中的0x11223344 为什么就不需要与0xff?

发现貌似是针对IP出现负数做的操作

[回复](#) | [举报](#)

fanxiao_cn 2013-04-26 05:36:18

大神，为什么“`printf("IP = %x.%x.%x.%x\n", *(c)&0xff, *(c+1)&0xff, *(c+2), *(c+3));`”里面*c和*(c+1)要&0xff,我试了不与的话像*c返回值会成为0xffffffffc0,为什么呢？第一个例子中的0x11223344 为什么就不需要与0xff?

[回复](#) | [举报](#)

评论热议

请登录后评论。

[登录](#) [注册](#)

[关于我们](#) | [关于IT168](#) | [联系方式](#) | [广告合作](#) | [法律声明](#) | [免费注册](#)

Copyright 2001-2010 ChinaUnix.net All Rights Reserved 北京皓辰网域网络信息技术有限公司. 版权所有

感谢所有关心和支持过ChinaUnix的朋友们

京ICP证041476号 京ICP证060528号