Netware API 的编程环境与系统功能调用

郭大伟　（安徽财贸学院计算中心）

李方平　（安徽省蚌埠市节能中心）

NOVELL 网络是目前流行的局域网,为鼓励网络应用的开发,并降低网络开发代价与简化开发过程,NOVELL 网络的操作系统 NetWare 提供了丰富的应用程序接口 API,利用这一特性使得第三方开发系统程序变得十分方便。本文拟简要介绍有关利用 NetWare 应用程序接口进行编程的环境与方法,供同行参考。

一、NetWare 的应用程序接口 API

DOS 的程序员一般对 DOS 的功能调用非常熟悉,与其类似,NetWare API 是一些事先定义好的、公用的、可随时调用的功能模块,应用程序可以通过使用 API 提供的系统服务,在较高层次和更抽象的环境中进行操作,要求处理的具体细节由 API 完成。网络操作系统 NetWare 利用 API,为其它应用程序提供了各种网络服务及功能调用,程序开发人员可以利用以下两种方式从 DOS 访问这些 NetWare API:

.DOS 环境下 NetWare C 语言接口

.DOS 环境下 NetWare 系统功能调用

利用这些开发工具,程序开发人员可开放式地访问 NetWare, 快速安全地实现文件管理、计帐、目录管理、通信、打印服务等,以增强网络功能。

我们知道,网络环境由两个基本部分组成,即文件服务器环境和工作站环境, 在工作站向网络注册时,必须执行 IPX.COM 和 NET3.COM 两个工作站外壳文件, 并驻留工作站内存。IPX.COM 提供网孙包交换协议支持,用于与文件服务器和其它工作站打交道 ,NET3.COM 是工作站外壳文件,是网络与 DOS 的接口。对 DOS 用户而言,NetWare 提供的编程接口主要由 IPX-NET3 这两个文件承担。从实现的角度来看,由于 NetWare 与 DOS 和 NETBIOS 兼容,所以在 MS-DOS 环境中,NetWare API 是作为中断处理程序而安装的,它通过设置中断向量并指向该中断处理向量,API 所占用的中断向量大部分是 MS-DOSINT 21H 中断调用接口的扩展,当工作站与网络通讯时,IPX 和 NET3 截取了一系列 DOS 中断,调用请求信息都直接交给 IPX-NET3 处理,并通过它们与文件服务交换信息。由于 API 只是扩展和增强了 MS-DOS,而不是取代 MS-DOS,所 NetWare API 允许在单用户 MS-DOS 环境下开发的软件不加修改地在 NetWare 下运行。

在网络环境中,基于网络分层的概念,API 的结构是分层组织的,API 提供了由最低层到最高层的软件分层结构,较低层的 API 提供基本服务,而较高层的的 API 用于完成较复杂的功能。正如 MS-DOS 的 DOS 中断程序可以频繁使用 ROM BIOS 一样,较高层的 API 函数可以使用较低层次的函数,并且不受低层函数实现方法的影响。

二、NetWare 系统开发的编程环境

为帮助用户快速有效地进行系统开发,NOVELL 提供了两个开发工具包:

1.NetWare C Interface for DOS

NetWare C Interface for DOS 是一组函数库,它使程序开发人员可使用 C 语言直接对 NetWare 系统进行调用,该工具包包括由四种 C 语言编译器(NOVELL 、WATC OM,Borland,Microsoft,Lattice)生成的函数库、资料、以及 C 程序源代码。这些库函数给出了丰富的 C 语言源程序调用实例 , 其函数的底层实现实际上都是调用了 NetWare 的系统接口,在使用该工具包时,必须配以相应的 C 语言编译系统。

2.NetWare System Call for DOS

NetWare System Call for DOS 为程序设计人员在网络环境下对 NetWare 系统功能调用提供了一套详细的文档,该文档列出 8086 汇编语言在网络系统一级进行功能调用的命令,提供了每个系统功能调用的描述和讨论、代码参数、实例程序以及有用的备注。用户若使用该工具包,则需要汇编语言编译系统,其工作站的操作系统应为 DOS2.0 以上版本。

如果用户没有以上所介绍的软件开发工具包,那么可借助于汇编语广言或高级语言利用 NetWare 提供的系统功能调用接口直接编写基本函数,它不需要 NetWare C 的函数库,只需按照 API 的中断接口调用规则,通过寄存器传递参数和存放返回参数。访问 NetWare API 一般借助采用汇编语言或某种高级语言实现,鉴于目录 C 语言的流行以及强大的功能和可移植性,所以采用 C 作为编程工具是适宜的。读者在本文的最后将看到利用 Turbo C 调用系统功能的一个实例。

三、NetWare 的系统功能调用

NetWare 的系统调用功能十分丰富,共有一百多种,有关网络功能的详细调用内容,用户可参考 Novell NetWare 功能调用手册。NetWare 系统功能按其功能的分类,可以分为以下几个方面:即网络环境功能调用、网络锁定功能调用、网络打印功能调用、网络通信功能调用、网络目录请求功能调用、网络注册功能调用以及网络装订文件功能调用等,在调用这些系统功能时,应注意以下几个问题:

1.网络功能的调用格式

网络功能调用以十六进制或十进制的功能号开始,然后给出功能名,功能的目的,功能的输入参数,功能的返回值或状态码。

2.网络功能调用的数据类型

网络功能调用的数据类型有如下四种定义:

BYTE 表示一个 8 位字节的数字;

WORD 表示一个按高低顺序的二字节数字(16 位);

LONG 表示一个按高低顺序的四字节数字(32 位);

NATIVE 表示一个按低高顺序的二字节数字(16 位)。

在采用 C 语言编程时可将这些类型表示为:

```
typedef unsigned char BYTE
typedef unsigned int WORD
typedef unsigned long LONG
```

3.数据包缓冲区

文件服务器在处理某些功能调用时,需指向被传送的两个缓冲区的指针。第一个缓冲区是请求数据包缓冲区,其缓冲区地址由寄存器 DS:SI 指出,它保存要传送到网络上的信息;第二个缓冲区是应答数据包缓冲区,其缓冲区地址由寄存器 ES:DI 指出,它存储来自网络的任何信息。

四、网络功能调用实例

作为网络调用示例,本文采用 Turbo C 2.0 设计了一个网络环境功能调用的实例程序,用户在 Novell 环境下运行此程序后,可获取注册工作站与文件服务器的联接信息:工作站逻辑站号与物理站号:工作站版本信息及服务器名字等信息。程序中对网络功能调用的入口参数及返回参数作了详细说明,给出了调用 NetWare API 的一般方法。用户参照网络功能调用手册不难编制出实现各种功能的网络应用程序。

```
/ * 网络环境功能调用示例程序 * /
#indlude <stdio.h>
#indlude <dos.h>
typedef unsigned char BYTE;
BYTE far * GetFileServerName(void);
BYTE far * ShellVersion();
int Getnum(void);
int AttachToFileServer();
main()
    {
    BYTE far * serverNameTable;
    BYTE far * Version;
    int station,result;
    AttachToFileServer();
    station = Getnum();
    printf("You are logged into station %d.\n",station);
    Version = ShellVersion();
    WordStationAddr();
    printf("Shell OS = %Fs\n",Version);
    printf("Shell Version = %Fs\n",Version+6);
    printf("Hardware Type = %Fs\n",Version+12);
    serverNameTable = GetFileServerName();
    printf(S'erver Name:%Fs\n",serverNameTable);
    getch();
    }
```

```
/ * 函数功能:返回工作站与服务器联接信息
     寄存器入口参数:
     AH = F1h
     AS = 0
     DL = 服务器连接 ID
     寄存器出口参数:
     AL = 结束码
     00H 成功
     F8H 已经连接到服务器 ·
     F9H 在服务器连接表中没有空项
     FAH SHELL 连接 ID 表中没在空项
     FCH 不认识的服务器
     FEH 服务器的 Bindery 库已经锁住
     FFH 服务器没有响应 * /
int AttachToFileServer()
{
int result,freeSlot = 1;
_AH = 0XF1;
_AL = 0;
_DL = freeSlot;
geninterrupt(0x21);
result = _AL;
     switch(result){
     case 0x00: printf("Sucessfully attached to file sever.\n");
     break;
     case 0xF8: printf("Already attached to server.\n");
     break;
     case 0xF9: printf("Server has no more free connection.");
     break;
     case 0XFA: printf("No more servers slots available.\n");
     break;
     case 0xFC: printf("File server unkon.\n");
     break;
     case 0XFE: printf("Server bindery locked.\n");
     break;
     case 0xFF: printf("No response form server \n");
     exit(1);
     default: printf("Error attaching to server,code = %X",result);
     }

/ * 函数功能:获取工作站逻辑站号
     入口参数
     AH = DCh
     返回参数:
     注:逻辑站号是根据注册的顺序赋给工作站的
        逻辑端口,如果用户重新注册后可能改变 *
int Getnum()
     {
     _AH = 0xDC;
     geninterrupt(0x21);
     return  AL;
     }
 / * 函数功能:获取工作物理站号
     寄存器入口参数:
     AH = EEh
     寄存器出口参数:
     CX,BX,AX = 物理地址(16 进制)
     注:物理站号是网络接口卡所安装的节点地址。
     除非改变网络地址设置,否则该地址保持不变。 * /
int WorkStationAddr()
     {
     int ax,bx,cx;
     _AH = 0XEE;
     geninterrupt(0x21);
     ax = _AX;
     bx = _BX;
     cx = _CX;
     printf("The WorkStation Address is %X%X%X \n",cx,bx,ax);
     }
 / * 函数功能:返回工作站版本信息
     寄存器入口参数:
      AH = EAh
     AS = 01
     ES:DI 应答缓冲区地址
位移     内容              类型
0        操作系统类型   BYTE
?        操作系统版本   BYTE
?        硬件类型         BYTE
BYTE far * ShellVersion()
     {
     _AH = 0xEA;
     _AL = 01;
     geninterrupt(0x21);
     return MK_FP( , DI);
     }
/ * 函数功能:返回指向文件服务器名字表的指针
     寄存器入口参数:
     AH = EFh
     AS = 4
     输出:ES:SI 指向的文件服务器名字表指针 * /
BYTE far * GetFileServerName(void)
     {
     _AH = 0XEF;
     _AL = 4;
     geninterrupt(0x21);
     return ML_FP(_ES,_SI);
     }
```