Research of Embedded Home Gateway Network Driver and Software Realization

Guang Dong[1], Luping Jiang[2], Xiaohan Zhang[3]

School of Computer Science & Technology, Changchun University of Science & Technology
7089 Weixing Road, Changchun 130022, China

[1] lightdg@yahoo.com.cn; [2]jiangluping@yahoo.com.cn; [3]zhangxiaohan@126.com

***Abstract*-This paper introduced the present research and development trend of intelligence home and home gateway, studied network driver of embedded Linux system, and designed function and software structure of home gateway network. It gave the method of network driver and the corresponding data structure, and introduced in detail the embedded Linux Socket programming and serial interface operation.**

Keywords-Intelligence Home; Home Gateway; Embedded Linux; Network Driver; Serial Interface Operation

I. INTRODUCTION

As the development of the 3rd Generation Telecommunication (3G) technology[1], Internet of Things (IOT) technology[2], three nets fusion technology[3], the integration of household appliances and network information technology, network electrical appliances and intelligence home[4] are more and more widespread in home and abroad. Home gateway is one of the areas the research focusing on [5].

Home gateway is an independent, intelligence, flexible, and standardized household network system interface unit. It can receive communication signal through Internet, WAP, telephone, mobile phone and a variety of external network, which send signals to the specific user equipment through home internal network, and send the corresponding signal back to external communications nodes to achieve the entire distance interaction process.

Remote monitoring, data acquisition and system reconstruction, etc have changed with the development and application of embedded Internet products[6]. On one hand, along with the rapid development of Microprocessor technology and the improvement of system design, the speed of CPU is continuously raised. RISC technology and virtual technology has a wide range of applications. External circuit implement has realized the internal integration. Microprocessor in intelligent household appliances, intelligent water meter, watt-hour meter and other intelligent home field has a wide range of applications[7]. On the other hand, with the development of embedded technology and all kinds of network information product appearance, it has become possible for the development of embedded home gateway.

In fact, the theory of home gate way has been published and the technical difficulties have been solved before. However, it always fails to spread into the ordinary user's family because of its particularity. First, price restricting the development is a primary problem. The prices of home gateway must be low for the vast number of users at family acceptance. Second, the installation and using of home gateway must be more convenient, and the operation must be intelligent. The operation requires "The fool". Third, the work must be stable which does not need too much maintenance because it is impossible to provide a network administrator for each family. Embedded Linux in these aspects is suitable for the need of home gateway [8].

II. NETWORK DRIVER OF EMBEDDED HOME GATEWAY

Embedded Linux is applied for software architecture design of home gateway and its software. The essential method of the network driver of home gateway, network driver structure, and data structure of the network driver are introduced.

A. Essential Network Driver Method

Network equipment provides some system access methods as an object. The methods with unified interfaces concealed hardware details, which makes a unified visit on various network devices and realizes the hardware independency[9].

1) Initialize:

The driver must have an initialization method. When the driver is loaded into system, the initialization routine will be called to fulfil the following work. Equipment detection is to detect whether the hardware exists based on the hardware features and decide whether the driver is activated. Hardware initialization and configuration is fulfilled in initialization, such as plug and play hardware (Linux kernel has good support on PnP functions to complete the function). After the negotiation and configuration, the resources can be taken up from the system. Some resources can be shared with other devices, such as interruption, but others such as IO, DMA cannot. The following work is to initialize the variable in device structure. Finally, the hardware starts to work.

2) Open:

The OPEN method is called in the network equipment driver, when the network equipment activates (when the equipment status by DOWN to UP).Therefore, in fact in the very many INITIALIZE works may be done here such as resource application and hardware activation. If DEV->OPEN returns NON-0(ERROR), state of the hardware is still DOWN. Another function of OPEN method is, if DEV->OPEN returns NON-0(ERROR), state of the hardware is still DOWN. OPEN method another action is. If the driver is loaded as a module, it needs to prevent the equipment in open state when the module is unloaded. MOD_INC_USE_ COUNT Macro needs to be called in OPEN method.

3) Close:

CLOSE method takes the opposite of operation to OPEN method, which can release some resources to reduce the burden of system. The CLOSE method is called when the device status switch from UP to DOWN. In addition, if the driver is loaded as a module, the CLOSE needs to call

MOD_DEC_USE_COUNT for reducing equipment cited times so that the driver can be unloaded. CLOSE method must return success (0 = SUCCESS).

4) Send (Hard_Start_Xmit):

All the network device drivers must have the send methods. When system calls driver's XMIT, data will be sent to a SK_BUFF structure. The general driver passes the data to hardware. Also there have some special equipment such as LOOPBACK to compose the data as a receive data to send back to the system, or DUMMY equipment will directly discards the data.

If sending successfully, SK_BUFF will be released in the HARD_START_XMIT and returns 0 (successfully). If the equipment is unable to process temporarily such as the hardware is busy, returns 1. Now, if DEV-> TBUSY is set for NON-0, the system knows the hardware is busy and waits until DEV->TBUSY resets for sending again. The task to set TBUSY 0 is completed generally by interrupt. The hardware will give an interrupt after completing the sending. Now TBUSY might be set 0, then it is called with MARK_BH () to inform system to send once more. When the sending is not successful, it may also not set DEV->TBUSY NON-0 so that system can attempt the resend unceasingly. If HARD_START_XMIT sending is not successful, it does not have to release SK_BUFF. Data sent in SK_BUFF data contains the frame head which the hardware needs. Therefore, in the sending methods, it does not need again to fill the hardware frame head, and the data may be submitted directly by hardware. SK_BUFF is locked to ensure that the other programs will not access it.

5) Reception:

The driver does not have a receive method. It is driver that informs the system when there has data received. Generally, when the device receives data, it will give an interrupt. In the interrupt processing program, the driver will apply a buffer SK_BUFF (SKB). Data are read from hardware and put into the applied buffer. Then fill the information to SK_BUFF. SKB-> DEV = DEV, and judge the protocol type of the received frame, fill in the SKB-> PROTOCOL (more protocol support). The pointer SKB->MAC.RAW is set to hardware data, and then discards hardware frame head (SKB_PULL). SKB->PKT_TYPE is also needed to be set for marking the second layer (link layer) data types. It can be the following types:

PACKET_BROADCAST: Link layer radio.

PACKET_MULTICAST: Link layer multicast.

PACKET_SELF: Send to own frames.

PACKET_OTHERHOST: The frame sent to others (listen mode will have this kind of frame). Finally call NETIF_RX () to send data to the protocol layer. NETIF_RX () put the data in processing queue and then return, and the real processing is after the interrupt returns, which can reduce interrupt time. After calling NETIF_RX (), the driver can't again access data buffer SKB.

6) Hard Header:

Usually, before upper data send, the hardware will add own hardware frame head, such as Ethernet have 14 bytes frame head. This frame head is added in the front of the upper level such as IP and IPX data packets. The driver provides a HARDE_HEADER method, and the protocol layer (IP, IPX ARP, etc) will call it before sending data.

The length of the hardware frame heads must be filled in DEV->HARD_HEADER_LEN. Protocol layer will reserve space of hardware frame heads so that HARDE_HEADER program only calls SKB_PUSH, then fills in the correct hardware frame head. When the protocol layer calls HARDE_HEADER, the sent parameters include (2.O.xx): SK_UFF, DEVICE pointer, PROTOCOL, the destination (DADDR), the source (SADDR) and the data length (LEN). The data's length will not use the parameters in SK_BUFF because when calling HARD_HEADER, the data possibly has not organized completely. If SADDR is NULL, it uses the default address (DEFAULT). DADDR is NULL means that protocol layer dose not know the hardware destination address. If HARD_HEADER completely fill in the hardware frame head, return the added bytes. If the information in the hardware frame header is not completely yet (for example, when DADDR is NULL but the frame header needs destination hardware address. Typically Ethernet needs to address analytical (APP)), and return the negative bytes.

In HARD_HEADER returns negative, the protocol layer can make further BUILD_HEADER work. At present, Linux system makes ARP (if HARDE_HEADER returns positively, DEV->ARP=1, it indicates no need to make ARP; if returns negative, DEV->ARP=0, then makes ARP). The HARDE_HEADER is called in each protocol layer of the processing procedure, such as IP_OUTPUT.

7) Address Resolution (Arp):

Some network has hardware address (such as Ethernet). When sending hardware frame, it needs to know the purpose hardware address, which needs the upper level protocol address (IP, IPX) to correspond with the hardware address. This kind of correspondence is fulfilled by ARP. The ARP device will call REBUILD_ HEADER before sending.

The main calling parameters include pointers of hardware frame head and protocol layer address. If the driver can resolve hardware address, return to 1; otherwise, return to 0. REBUILD_ HEADER calls are in the DO_ DEV_QUEUE_XMIT () of the NET/CORE/DEV.C.

8) Parameter Setting and Counting Data:

The driver also provides some methods for setting system device parameters and reading information. Usually, only super users (ROOT) can set the device parameters. The setting methods are:

DEV-> SET_MAC_ADDRESS ()

When user calls IOCTL and the type is SIOCSIFHWADDR, it means to set the device's MAC address. Generally, the setting of MAC address has no too much significance.

DEV-> SET_CONFIG ()

When user calls IOCTL and the type is SIOCSIFMAP, the system will call the driver SET_CONFIG method, and the user will deliver the IFMAP structure including parameters such as I/O, interrupt.

DEV -> DO_IOCTL ()

If the user calls IOCTL and the type is between SIOCDEV and SIOCDEVPRIVATE + 15, the system will call the corresponding method of the driver, which is generally set the special data of the device.

Reading information is also through calling IOCTL. In addition that the driver can also provide a DEV->GET_STATS method, for returning an ENET_STATISTICS structure, which contains the counts information of sending and receiving.

The OCTL processing is included in DEV_IOCTL () and DEV_IFSIOC () of the NET/CORE/DEV.C.

B. *Structure of Network Driver*

All the Linux network drivers follow the generic interface[10]. The design uses object-oriented method. A device is an object (DEVICE structure), inside which it has its own data and methods. When each equipment method is called, first parameter is the device object itself. So this method can access own data (similar to object-oriented program design of "THIS" reference). The essential methods of a network device include initialization, sending, and reception. Initialization routine completes the hardware and DEVICE variable initialization and system resources application. The sending routine is automatically called when the driver's upper level protocol layer has the requirements to send data. The general driver does not provide buffer for the send data, but uses the hardware transmission function to send directly the data. The reception is informed by hardware interrupt. In the interrupt handlers, the hardware frame information will be filled in a SK_BUFF structure, and then NETIF_RX () will be called to transfer to the upper level processing.

C. *Data Structure of Network Driver*

In the network drivers, the most important thing is the data structure of network equipment.

It is defined in the INCLUDE/LINUX/ NETDEVICE.H. As shown in the following.

Struct_evice

{

……

/* Low level status flags.*/

volatile unsigned char start, /*start an operation*/

volatile unsigned int interrupt; /*interrupt arrived*/

/*When handling interrupts interrupt set to 1, processes the reset.*/

unsigned long busy; /*transmitter busy must belong for bit ops*/

struct_device *next; /*the device initialization function. Called only once.*/

/*Point to the driver's initial method.*/

int (*init)(struct device *dev); /*Some hardware also needs these fields, but they are not part of the usual set specified in Space. c. */

/*Some hardware, can be in a board support multiple interface, if_port may be used.*/

unsigned char if port; /*Select able AUI,TP,..*/

unsigned char dma; /*DMA channel*/

struct enet_statistics *(*get_stats)(struct device*dev);

/*This marks the end of the "visible" part of the structure. All fields here after are internal to the system, and may change at*will (read: maybe cleaned up at will). These maybe needed for future network-power-down code trans-start. The last record the time successfully sent. Can be used to determine the hardware is working correctly. */

unsigned long trans_start; /*Time (in jiffies) of last Tx*/

unsigned long last_ rx; /*Time of last Rx*/

/*in the flags there are a lot of content, defined in include/linux/if.h.*/

unsigned short flags; /*interface flags (ala BSD)*/

unsigned short family; /*address family ID(AF_NET)*/

unsigned short metric; /*routing metric (not used)*/

unsigned short mtu; /*interface MTU value*/

/*type Indicate the type of physical hardware. The main hardware that whether need arp. Defined in include / linux / if_arp.h.*/

unsigned short type; /*interface hardware type*/

/*The upper protocol layer, according to hard_header_len, in front of the sending data buffer, reserves a hardware frame space.*/

unsigned shorthard header len; /*hardware head length*/

/*priv point to drivers from the definition of some parameters.*/

void *priv; /*pointer to private data*/

/*Interface address info.*/

……

};

III. SOFTWARE REALIZATION OF EMBEDDED HOME GATEWAY

A. *Home Gateway Function and Principle*

The home gateway is the intelligent plot terminal device and the core, its main functions and principles are shown in Fig. 1.

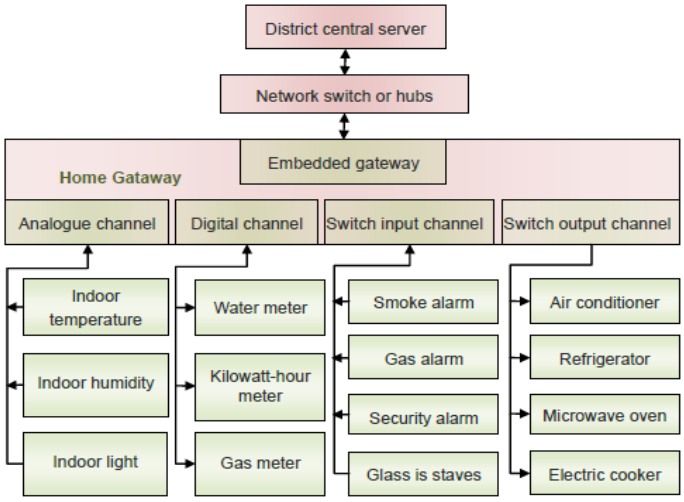

Fig. 1 Home gateway function and principle

Fig. 1 shows that, intelligent home equipments are connected through the home gateway to provide interface home gateway, and connected to the network switch or hubs through the embedded gateway, so as to realize communication between intelligent home equipment and the district central server.

B. Software Architecture of Embedded Home Gateway

Embedded Home Gateway mainly realizes the data exchange of major internal RS232 and external internet data. When the gateway receives an access requesting from the remote host, it reads from the RS-232 serial, does the corresponding processing, calls kernel, BSD socket, the transport layer and network layer by Linux system, and adds the corresponding logical addresses and other data for the network layer. Then IP datagram is encapsulated, a physical address is added to the MAC layer and other corresponding MAC frame data are added to network card. Finally, the data are sent by hardware. The software structure is shown as Fig. 2.

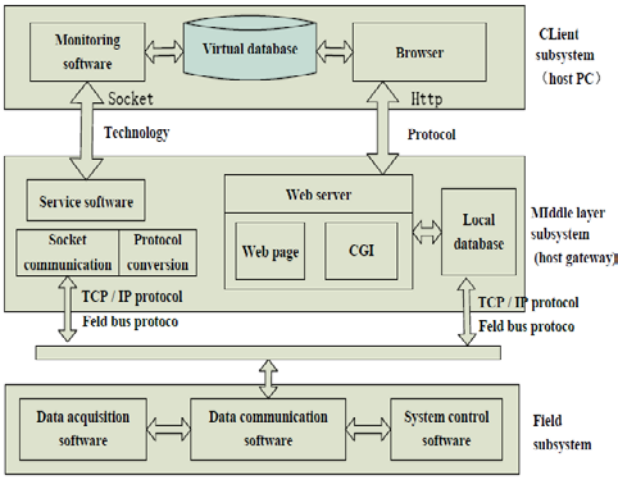

Fig. 2 Software architecture of embedded home gateway

C. RS -232 Serial Interface Operations

RS-232 is a standard serial interface. It is commonly used in computers, peripherals, switches and many other communication equipments[11]. Linux provides good support on RS-232 from the beginning.

The head files needed in Linux serial port operating are:

```
# include <stdio.h> /*standard input/output definition * /
# include <stdlib.h> /*standard function library definitions */
# include <unistd.h> /*Unix standard function define * /
# include <sys/types.h>
# include <sys/stat. h>
# include <fcntl.h> /* files control defines */
# include <errno.h> /* error code definition */
# include <termios.h> /* PPSIX terminal control defines* /
```

Serial port files of Linux are /dev/ttyS0 and /dev/ttyS1 in /dev. Its basic operations include opening serial port, setting serial interface, reading serial port, writing serial port and closing serial port. The basic serial port settings includes baud rate setting, check bit setting and stop bit setting. The serial

port settings will mainly set the various member values of structure STRUCT_TERMIOS:

```
Struct_termios
{
unsigned short c_iflag; /* input pattern flag */
unsigned short c_oflag; /* output pattern flag */
unsigned short c_cflag; /* control pattern flag */
unsigned short c_lflag; /* local pattern flag */
unsigned char c_ine ; /*linediscipline*/
unsigned char c_cc[NCC]: /* special control character */
}
```

The five members in STRUCT_TERMIOS structure correspond to the input, output, control, local patterns and special characters of terminals. Terminal access control function is shown in Table I.

TABLE I
TERMINAL ACCESS CONTROL FUNCTION

Function	Description
tcgetattr	Get terminal attributes (termios structure)
tcsetarrt	Set terminal attributes (termios structure)
cfgetispeed	Get input rate
cfgetospeed	Get output rate
cfsetispeed	Set input rate
cfsettospeed	Set output rate
tcdrain	Waiting for all output was transmitted
fclow	Suspend sending or receiving
toflush	Forsake the queue has not send or receive data
tcsendbreak	Send BREAK characters
tcgetgrp	Get foreground process group ID

The embedded serial communication of Linux is illustrated as follows:

Step1. Open the device file by OPEN. Open flag O_NONBLOCK and O_NOCCTY will be used. O_ONBLOCK will return immediately without waiting, and O_OCCTY specifies that the serial device is not control terminal;

Step2. Set communication baud rate by CFSETISPEED and CFSETOSPEED;

Step3. Set odd parity bit;

Step4. Read or write serial port;

Step5. Read or write end, and close the serial port by the CLOSE.

D. Socket Programming in Embedded Linux

Network communication is a basic function of the gateway service program. The network communication of Linux system is evolved from UlVIX4.3BSD model, which

supports all the BSD Socket and TCP/IP functions. Linux supports multiple Sockets, so that it is suitable for both the general inter-process communication and the inter-process communication in network environment. Socket has logically three elements: domain, type, and procedures. Domain shows which kind of network the socket are used. Type means the network communication pattern, which includes connection oriented and connectionless two means of communication. Combining with domain and type can identify the applicable procedures in general. For example domain for AF_INET, type has the connection, the procedure is basically TCP. The process with socket communication uses server/client mode. Linux kernel provides a unified system called interface SYS_SOCKET_CALL (int call, unsigned long * args) for all the operations related with Socket. The first parameter call is the specific OPCODE defined in the INCLUDE/LINUX/NET. H. The common BSD Socket API [12] is introduced as follows:

INT socket (int domain, int type, int protocol) is used to get socket descriptors. The first parameter is set to AF_INET; the second parameter is interface type: SOCK_STREAM or SOCK_DGRAM, and the third parameter is 0 according to the first two parameters procedures.

For the servers, the next step calls BIND () to bind to a port. The client calls CONNECT () through the system to set up a connection, and the server will wait for a connection request and deals with them. So it must first transfer LISTEN (), then transfer ACCEPT () to realize again. To send and receive data, we can generally use function SEND ()/RECV (), SENDTO ()/REOVFROM () and SENDMSG ()/RECVMSG (). It can also operate as normal as file reading and writing a socket. Finally the client and the server can call CLOSE () to shut down a socket. SELECT () is a very useful system call. It can monitor a few sockets simultaneously to determine which socket can read data, which can write data. It can also specify the waiting time to prevent the system from system call blocking. This function can also be used for testing equipment of other types. The function is as follows:

int select(int numfds, fd set *readfds, fd set*writefds, fd set *exceptfds, struct timeval *timeout);

If it needs to read data of the descriptor, add only the descriptor to READFDS, and at the same time NUMFDS is set as the greatest file descriptors value plus one. SELECT () will modify the READFDS (). It can use FD_ISSET () to test which descriptor has been ready for reading. The write operation of descriptors is similar. The basic set of socket functions is shown in Fig. 3.

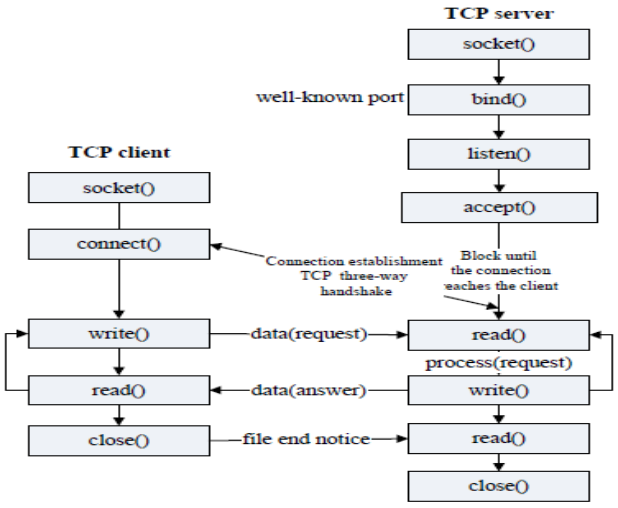

Fig. 3 Basic TCP/IP Socket function

IV. CONCLUSIONS

The intelligence home unifies computer technology, computer network technology, embedded system technology, network communication, signal processing technology, long distance control technology, and intelligent control technology. Home gateway research plays a powerful role in the development of intelligence homes.

This paper studies the development method of home gateway network driver and data structure. It gives a communication principle diagram and home gateway function structure, and designs the embedded home gateway software structure diagram. Finally, it discusses in details RS-232 serial interface communication under embedded Linux environment and the embedded Linux Socket programming. It establishes the embedded Linux development environment on VMware Workstation, debugs and tests the network driver for home gateway, and realizes socket programming in RS-232 serial port network communication and embedded Linux system. It obtains the good effect.

ACKNOWLEDGMENT

Here, we would like to give thanks to our professions who gave us support and advice, especially Dr. Wei He in School of Computer Science and Technology of Changchun University of Science and Technology.

REFERENCES

[1] Jun Wang. "Gateway Technology Applications in 3G Mobile Communication System", *Communications Technology*, vol. 44, pp. 120-121, Jun. 2011.

[2] Jianwu Zhang, Huan Yan, Jianrong Bao. "Design of Intelligent Family Gateway and Its Application in Internet of Things", *Computer Engineering*, vol. 37, pp. 246-248, Sep. 2011.

[3] Fan Li. "Home gateway prospect forecast", *China Telecommunications Trade*, vol. 3 pp. 123. Mar. 2011.

[4] Mingjie Zhang, "Analysis and Design of Home Gateway in Smart Home System", *Science Technology and Engineering*, vol. 9, pp. 1921-1924, Apr. 2009.

[5] Ling Zhang, Dengji Hu, Yongjin Xu. "Gateway Study for Intelligent Home System", *Instrument Technology*, vol. 3, pp. 15-17, Mar. 2011.

[6] Wenzao Shi, Ping Wang, Xi Huang, Wude Ye. "Software Design of Embedded Smart Home Gateway", *Computer System Application*, vol. 19, pp. 47-51, Oct. 2010.

[7] Cheng Chen, Xiaowei Qin, Ding Tang. "Research and Development of Sensor Network Accessing Module in Home Gateway", *Computer System Application*, vol. 20, pp. 45-48, Oct. 2011.

[8] Hongru Chen, Guoming Sang. "Research and Application of Embedded Wireless Home Gateway", *Computer & Digital Engineering*, vol. 39, pp. 165-169, Nov. 2011.

[9] Xuguang Li. ARM application system development explanation. Beijing, China: Tsinghua University Press, 2003.

[10] Haiyan Xu, Yan Fu. Embedded system technology and application. Beijing, China: Mechanic Industry Press, 2002.

[11] Lingyun Zhu, Kaiqi Cao, Zhao Chen. "Development of meter monitoring system based on embedded home gateway", *Micro Computer Information*, vol. 26, pp. 42-44, Nov. 2010.

[12] *ARM9TDMI (Rev4) Technical Reference Manual (ARM DDI0192A)*, ARM Limited, 2000.

Guang Dong was born in Huadian of China in December 1963. He graduated from Dalian University of Science and Technology of China in July 1985. He was in the department of computer science and engineering as a computer science professional and obtained technology bachelor's degree.

He is currently an associate professor, and works at School of Computer Science and Technology in Changchun University of Science and Technology, Changchun, China. He mainly engages in computer science and technology, embedded system and database technology application of teaching and research work.

Prof. Dong is a member of Chinese Computer Academic Society.

Luping Jiang was born in Changchun of China in September 1969. She graduated from Changchun University in July 1989. He was in the department of computer as a computer application professional, and obtained technology bachelor's degree.

She is currently an engineer, and works at School of Computer Science and Technology in Changchun University of Science and Technology, Changchun, China. She mainly engages in database technology application and embedded system of teaching and research work

Xiaohan Zhang was born in Changchun of China in July 1973. In July 1973, he was born in the Changchun of China, in July 1993 graduated from Changchun University of Science and Technology, the department of the electronic, the application electronic professionals, and obtains the technology bachelor's degree.

He is currently an engineer, and works at School of Computer Science and Technology in Changchun University of Science and Technology, Changchun, China. He mainly engages in computer science and technology, embedded system and database technology application of teaching and research work.