7.2 Introduction to Embedded Linux Programming

Disclaimer: The following example for Linux programs are FriendlyARM original, we know some companies or individuals to modify development board a copyright note, for themselves, although the domestic base for such plagiarism is not legally binding, but we are such a shameless theft to be despised, and advising everyone to respect the original manufacturer of hard labor.

7.2.1 LED test program

Source code description:

Driver source code directory	/opt/FriendlyARM/mini2440/linux2.6.32.2/drivers/char
Driver Name	mini2440_leds.c
Device Type	misc
Device Name	/dev/leds
Test program source code directory	/opt/FriendlyARM/mini2440/examples/leds
Test program source code name	led.c
Test program executable file name	led
Note: LED driver has been compiled into the default kernel, so cannot be loaded using insmod.	

Test program source code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/ioctl.h>
int main(int argc, char **argv)
{
int on;
int led_no;
int fd;
/ * Check led control of two parameters, if no parameter input, then
exit. * /
if (argc != 3 || sscanf(argv[1], "%d", &led_no) != 1 ||
sscanf(argv[2],"%d", &on) != 1 ||
on < 0 || on > 1 || led_no < 0 || led_no > 3) {
      fprintf(stderr, "Usage: leds led_no 0|1\n");
      exit(1);
      }
/ * Open / dev / leds device file * /
fd = open("/dev/leds0", 0);
```

```
if (fd < 0) {
      fd = open("/dev/leds", 0);
      }
if (fd < 0) {
      perror("open device leds");
      exit(1);
      }
/ * Ioctl system calls and enter through the parameters of control
led * /
ioctl(fd, on, led_no);
/ * Close the device handle * /
close(fd);
return 0;
}
```

7.2.2 Button test program

Source code description:

Driver source code directory	/opt/FriendlyARM/mini2440/linux-2.6.32.2/drivers/char
Driver Name	mini2440_buttons.c
Device Type	misc
Device Name	/dev/buttons
Test program source code directory	/opt/FriendlyARM/mini2440/examples/buttons
Test program source code name	buttons_test.c
Test program executable file name	buttons
Description: The button driver has been compiled into the default kernel, so cannot be loaded using insmod.	

Test program source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
int main(void)
{
int buttons_fd;
char buttons[6] = {'0', '0', '0', '0', '0', '0'};
buttons_fd = open("/dev/buttons", 0);
if (buttons_fd < 0) {
```

```
      perror("open device buttons");
      exit(1);
      }
for (;;) {
      char current_buttons[6];
      int count_of_changed_key;
      int i;
      if (read(buttons_fd, current_buttons, sizeof current_buttons)
!= sizeofcurrent_buttons)
      {
            perror("read buttons:");
            exit(1);
      }
      for (i = 0, count_of_changed_key = 0; i < sizeof buttons /
sizeof buttons[0]; i++) {
            if (buttons[i] != current_buttons[i]) {
                  buttons[i] = current_buttons[i];
                  printf("%skey %d is %s", count_of_changed_key? ", ":
"", i+1, buttons[i] == '0' ? "up" : "down");
                  count_of_changed_key++;
            }
      }
      if (count_of_changed_key) { printf("\n"); }
}
close(buttons_fd);
return 0;
}
```

You can follow the steps above, the hello program buttons to manually compile the executable file, and then downloaded to the development board to run it.

7.2.3 PWM control buzzer test program

Source code description:

Driver source code directory	/opt/FriendlyARM/mini2440/linux-2.6.32.2/drivers/char
Driver Name	mini2440_pwm.c
Device Type	misc
Device Name	/dev/pwm
Test program source code directory	/opt/FriendlyARM/mini2440/examples/pwm
Test program source code name	pwm_test.c
Test program executable file name	pwm_test
Note: PWM control buzzer driver has been compiled into the default kernel, so cannot be loaded using insmod.	

Test program source code:

```
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <stdlib.h>
#define PWM_IOCTL_SET_FREQ 1
#define PWM_IOCTL_STOP 2
#define ESC_KEY 0x1b

static int getch(void)
{
	struct termios oldt,newt;
	int ch;
	if (!isatty(STDIN_FILENO)) {
		fprintf(stderr, "this problem should be run at a
terminal\n");
		exit(1);
	}

	// save terminal setting
	if(tcgetattr(STDIN_FILENO, &oldt) < 0) {
		perror("save the terminal setting");
		exit(1);
	}

	// set terminal as need
	newt = oldt;
	newt.c_lflag &= ~( ICANON | ECHO );
	if(tcsetattr(STDIN_FILENO,TCSANOW, &newt) < 0) {
		perror("set terminal");
		exit(1);
		}

	ch = getchar();

	// restore termial setting
	if(tcsetattr(STDIN_FILENO,TCSANOW,&oldt) < 0) {
		perror("restore the termial setting");
		exit(1);
	}
	return ch;
}

static int fd = -1;
static void close_buzzer(void);

static void open_buzzer(void)
{
	fd = open("/dev/pwm", 0);
	if (fd < 0) {
		perror("open pwm_buzzer device");
		exit(1);
	}
```

```
	// any function exit call will stop the buzzer
	atexit(close_buzzer);
}

static void close_buzzer(void)
{
	if (fd >= 0) {
		ioctl(fd, PWM_IOCTL_STOP);
		close(fd);
		fd = -1;
	}
}

static void set_buzzer_freq(int freq)
{
	// this IOCTL command is the key to set frequency
	int ret = ioctl(fd, PWM_IOCTL_SET_FREQ, freq);
	if(ret < 0) {
		perror("set the frequency of the buzzer");
		exit(1);
	}
}

static void stop_buzzer(void)
{
	int ret = ioctl(fd, PWM_IOCTL_STOP);
	if(ret < 0) {
		perror("stop the buzzer");
		exit(1);
	}
}

int main(int argc, char **argv)
{
	int freq = 1000 ;
	open_buzzer();
	printf( "\nBUZZER TEST ( PWM Control )\n" );
	printf( "Press +/- to increase/reduce the frequency of the
BUZZER\n" ) ;
	printf( "Press 'ESC' key to Exit this program\n\n" );

	while( 1 )
	{
		int key;
		set_buzzer_freq(freq);
		printf( "\tFreq = %d\n", freq );
		key = getch();
		switch(key) {
			case '+':
				if( freq < 20000 )
					freq += 10;
				break;
			case '-':
				if( freq > 11 )
					freq -= 10 ;
```

```
                    break;
                case ESC_KEY:
                case EOF:
                    stop_buzzer();
                    exit(0);
                default:
                    break;
            }
        }
}
```

7.2.4 I2C-EEPROM test program

Source code description:

Driver source code directory	/opt/FriendlyARM/mini2440/linux2.6.32.2/drivers/i2c/busses
Driver Name	I2c-s3c2410.c
Device Type	Character device
Device Name	/dev/i2c/0
Test program source code directory	/opt/FriendlyARM/mini2440/examples/i2c
Test program source code name	eeprog.c 24cXX.c
Test program executable file name	I2c
Note: I2C drivers have been compiled into the default kernel so can't be loaded using insmod.	

Test program source code:

Note: The following program need to process the same directory 24cXX.c support

```
/*************************************************************************
********
            copyright : (C) by 2009 Guangzhou FriendlyARM, in China
            email : capbily@163.com
            website : http://www.arm9.net
*************************************************************************
*******/
#include <stdio.h>
#include <fcntl.h>
#include <getopt.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```
#include "24cXX.h"

#define usage_if(a) do { do_usage_if( a , __LINE__); } while(0);
void do_usage_if(int b, int line)
{
      const static char *eeprog_usage =
            "I2C-24C08(256 bytes) Read/Write Program, ONLY FOR
TEST!\n"
            "FriendlyARM Computer Tech. 2009\n";
      if(!b)
            return;
      fprintf(stderr, "%s\n[line %d]\n", eeprog_usage, line);
      exit(1);
}
#define die_if(a, msg) do { do_die_if( a , msg, __LINE__); }
while(0);
void do_die_if(int b, char* msg, int line)
{
      if(!b)
            return;
      fprintf(stderr, "Error at line %d: %s\n", line, msg);
      fprintf(stderr, " sysmsg: %s\n", strerror(errno));
      exit(1);
}
static int read_from_eeprom(struct eeprom *e, int addr, int size)
{
      int ch, i;
      for(i = 0; i < size; ++i, ++addr)
      {
            die_if((ch = eeprom_read_byte(e, addr)) < 0, "read
error");
            if( (i % 16) == 0 )
                  printf("\n %.4x| ", addr);
            else if( (i % 8) == 0 )
                  printf(" ");
                  printf("%.2x ", ch);
                  fflush(stdout);
      }
      fprintf(stderr, "\n\n");
      return 0;
}

static int write_to_eeprom(struct eeprom *e, int addr)
{
      int i;
      for(i=0, addr=0; i<256; i++, addr++)
      {
            if( (i % 16) == 0 )
                  printf("\n %.4x| ", addr);
            else if( (i % 8) == 0 )
                  printf(" ");
                  printf("%.2x ", i);
                  fflush(stdout);
                  die_if(eeprom_write_byte(e, addr, i), "write
error");
```

```
	}
	fprintf(stderr, "\n\n");
	return 0;
}

int main(int argc, char** argv)
{
	struct eeprom e;
	int op;

	op = 0;

	usage_if(argc != 2 || argv[1][0] != '-' || argv[1][2] !=
'\0');
	op = argv[1][1];

	fprintf(stderr, "Open /dev/i2c/0 with 8bit mode\n");
	die_if(eeprom_open("/dev/i2c/0", 0x50, EEPROM_TYPE_8BIT_ADDR,
&e) < 0,
		"unable to open eeprom device file "
		"(check that the file exists and that it's readable)");

	switch(op)
	{
	case 'r':
		fprintf(stderr, " Reading 256 bytes from 0x0\n");
		read_from_eeprom(&e, 0, 256);
		break;
	case 'w':
		fprintf(stderr, " Writing 0x00-0xff into 24C08 \n");
		write_to_eeprom(&e, 0);
		break;
	default:
		usage_if(1);
		exit(1);
	}

	eeprom_close(&e);

	return 0;
}
```

7.2.5 Serial test program

Source code description:

Driver source code directory	/opt/FriendlyARM/mini2440/linux2.6.32.2/drivers/serial/
Driver Name	S3c2440.c
Device Name	/dev/ttySAC0, /dev/ttySAC1, /dev/ttySAC2
Test program source code directory	/opt/FriendlyARM/mini2440/examples/comtest
Test program source code name	comtest.c
Test program executable file name	armcomtest
Description: The test program compile x86 version and arm version available, its source code is exactly the same.	

Note: comtest program is FriendlyARM of a serial port test program, serial terminal program is very simple and similar to Linux in minicom, the program manage with the hardware, so the same code applies only to any ARM Linux development board platform, you can also run on the PC Linux use, methods are exactly the same.


```
# include <stdio.h>
# include <stdlib.h>
# include <termio.h>
# include <unistd.h>
# include <fcntl.h>
# include <getopt.h>
# include <time.h>
# include <errno.h>
# include <string.h>
static void Error(const char *Msg)
{
      fprintf (stderr, "%s\n", Msg);
      fprintf (stderr, "strerror() is %s\n", strerror(errno));
      exit(1);
}

static void Warning(const char *Msg)
{
      fprintf (stderr, "Warning: %s\n", Msg);
}

static int SerialSpeed(const char *SpeedString)
```

```
{
      int SpeedNumber = atoi(SpeedString);
      #define TestSpeed(Speed) if(SpeedNumber==Speed) return
B##Speed
      TestSpeed(1200);
      TestSpeed(2400);
      TestSpeed(4800);
      TestSpeed(9600);
      TestSpeed(19200);
      TestSpeed(38400);
      TestSpeed(57600);
      TestSpeed(115200);
      TestSpeed(230400);
      Error("Bad speed");
      return -1;
}

static void PrintUsage(void)
{
      fprintf(stderr, "comtest -interactive program of com port\n");
      fprintf(stderr, "press [ESC] 3 times to quit\n\n");
      fprintf(stderr, "Usage: comtest [-d device] [-t tty] [-s
speed] [-7] [-c] [-x] [-o] [-h]\n");
      fprintf(stderr, " -7 7 bit\n");
      fprintf(stderr, " -x hex mode\n");
      fprintf(stderr, " -o output to stdout too\n");
      fprintf(stderr, " -c stdout output use color\n");
      fprintf(stderr, " -h print this help\n");
      exit(-1);
}

static inline void WaitFdWriteable(int Fd)
{
      fd_set WriteSetFD;
      FD_ZERO(&WriteSetFD);
      FD_SET(Fd, &WriteSetFD);
      if (select(Fd + 1, NULL, &WriteSetFD, NULL, NULL) < 0) {
            Error(strerror(errno));
      }
}
int main(int argc, char **argv)
{
      int CommFd, TtyFd;

      struct termios TtyAttr;
      struct termios BackupTtyAttr;

      int DeviceSpeed = B115200;
      int TtySpeed = B115200;
      int ByteBits = CS8;
      const char *DeviceName = "/dev/ttyS0";
      const char *TtyName = "/dev/tty";
      int OutputHex = 0;
      int OutputToStdout = 0;
      int UseColor = 0;
```

```
opterr = 0;
for (;;) {
      int c = getopt(argc, argv, "d:s:t:7xoch");
      if (c == -1)
            break;
      switch(c) {
            case 'd':
                  DeviceName = optarg;
                  break;
            case 't':
                  TtyName = optarg;
                  break;
            case 's':
                  if (optarg[0] == 'd') {
                        DeviceSpeed = SerialSpeed(optarg + 1);
                  }
                  else if (optarg[0] == 't') {
                        TtySpeed = SerialSpeed(optarg + 1);
                  }
                   else
                        TtySpeed=DeviceSpeed=SerialSpeed(optarg);
                  break;
            case 'o':
                  OutputToStdout = 1;
                  break;
            case '7':
                  ByteBits = CS7;
                  break;
            case 'x':
                  OutputHex = 1;
                  break;
            case 'c':
                  UseColor = 1;
                  break;
                  case '?':
                  case 'h':
                  default:
                  PrintUsage();
            }
      }
      if (optind != argc)
            PrintUsage();

      CommFd = open(DeviceName, O_RDWR, 0);
      if (CommFd < 0)
            Error("Unable to open device");
      if (fcntl(CommFd, F_SETFL, O_NONBLOCK) < 0)
            Error("Unable set to NONBLOCK mode");
      memset(&TtyAttr, 0, sizeof(struct termios));
      TtyAttr.c_iflag = IGNPAR;
      TtyAttr.c_cflag=DeviceSpeed|HUPCL|ByteBits|CREAD|CLOCAL;
      TtyAttr.c_cc[VMIN] = 1;
      if (tcsetattr(CommFd, TCSANOW, &TtyAttr) < 0)
            Warning("Unable to set comm port");
```

```
            TtyFd = open(TtyName, O_RDWR | O_NDELAY, 0);
            if (TtyFd < 0)
                  Error("Unable to open tty");
            TtyAttr.c_cflag=TtySpeed|HUPCL|ByteBits|CREAD|CLOCAL;
            if (tcgetattr(TtyFd, &BackupTtyAttr) < 0)
                  Error("Unable to get tty");
            if (tcsetattr(TtyFd, TCSANOW, &TtyAttr) < 0)
                  Error("Unable to set tty");
            for (;;) {
                  unsigned char Char = 0;
                  fd_set ReadSetFD;

                  void OutputStdChar(FILE *File) {
                        char Buffer[10];
                        int Len = sprintf(Buffer, OutputHex ? "%.2X "
: "%c", Char);
                        fwrite(Buffer, 1, Len, File);
                  }
                  FD_ZERO(&ReadSetFD);
                  FD_SET(CommFd, &ReadSetFD);
                  FD_SET( TtyFd, &ReadSetFD);
                  # define max(x,y) ( ((x) >= (y)) ? (x) : (y) )
                  if (select(max(CommFd, TtyFd) + 1, &ReadSetFD, NULL,
NULL, NULL) < 0) {
                  Error(strerror(errno));
            }
            # undef max
            if (FD_ISSET(CommFd, &ReadSetFD)) {
                  while (read(CommFd, &Char, 1) == 1) {
                        WaitFdWriteable(TtyFd);
                        if (write(TtyFd, &Char, 1) < 0) {
                              Error(strerror(errno));
                        }
                        if (OutputToStdout) {
                              if (UseColor)
                                    fwrite("\x1b[01;34m", 1, 8,
stdout);
                                    OutputStdChar(stdout);
                              if (UseColor)
                                    fwrite("\x1b[00m", 1, 8, stdout);
                                    fflush(stdout);
                        }
                  }
            }
            if (FD_ISSET(TtyFd, &ReadSetFD)) {

                  while (read(TtyFd, &Char, 1) == 1) {
                        static int EscKeyCount = 0;
                        WaitFdWriteable(CommFd);

                        if (write(CommFd, &Char, 1) < 0) {
                              Error(strerror(errno));
                        }

                        if (OutputToStdout) {
```

```
                              if (UseColor)
                                   fwrite("\x1b[01;31m",1,8,stderr);
                                   OutputStdChar(stderr);
                              if (UseColor)
                                   fwrite("\x1b[00m",1,8,stderr);
                                   fflush(stderr);
                         }

                         if (Char == '\x1b') {
                              EscKeyCount ++;
                              if (EscKeyCount >= 3)
                              goto ExitLabel;
                         }

                          else
                              EscKeyCount = 0;
                         }
               }
          }

ExitLabel:
          if (tcsetattr(TtyFd, TCSANOW, &BackupTtyAttr) < 0)
               Error("Unable to set tty");

          return 0;
}
```

7.2.6 UDP network test program

Source code description:

Driver source code directory	/opt/FriendlyARM/mini2440/linux-2.6.32.2/drivers/net/
Driver Name	dm9000.c
The driver major number	-
Device Name	eth0 (network equipment is not in the /dev directory)
Test program source code directory	/opt/FriendlyARM/mini2440/examples/udptak
Test program source code name	udptalk.c
Test program executable file name	udptalk.c
Description: The test program compile x86 version and arm version available, its source code is exactly the same.	

TCP/IP is protocol in transport layer: UDP (User Datagram Protocol). UDP and TCP are very different since connectionless socket programming and connection the socket programming because connectionless has sending a packet received contains the sender and recipient address information.

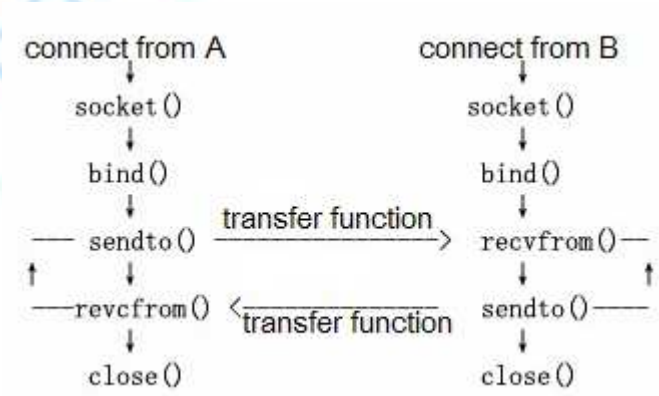
Before sending and receiving data, first create a packet socket means, the socket is of type SOCK_DGRAM, use the following call produces:

```
sockfd=socket(AF_INET, SOCK_DGRAM, 0);
```

As the need to establish a connection, resulting in socket can send and receive after had. Of course to receive a datagram must also bind a port, or the sender can't know which port to send, sendto and recvfrom two system calls are used to send and receive packet, the call format.

```
int sendto(int s, const void *msg, int len, unsigned int
flags, const struct sockaddr *to, int tolen);
int recvfrom(int, s, void *buf, int len, unsigned int flags,
struct sockaddr *from, int fromlen);
```

Where s is the use of socket, msg and buf are send and receive buffer pointer, len is the buffer length, flags for the option flag, here is also less than, is set to 0. is sent to and from the destination address and receive the source address contains the IP address and port information. The tolen and fromlen are both to and from the length of socket address structure. Both the return value is the actual number of bytes sent and received, return -1 indicating an error. No connection with the basic communication process as shown.



The basic process of the UDP communication.

The figure describes the communication port address both bind their case, but in some cases one party may not bind address. Since not bind the party's address and port allocated by the kernel, can't know in advance because the other party is not bound to the port and IP address (assuming the host has multiple ports that are assigned a different IP address), it is not bound only by the party issuing the first packet, the other based on income to the source of the data reported in the return address can send packet to determine the need to send address clearly. In this case the other party must be bound address port, and communication can only be initiated by a non-binding side.

And read() and write() is similar to the process block in recvfrom() and sendto() also occur. But in different ways with the TCP is the number of bytes received a packet 0 is possible, the application can be sendto() the msg is set to NULL, while len set to 0.

Following is a principle based on the above analysis of a UDP programming example:

```
/*
* udptalk: Example for Matrix V; Note: This procedure also applies
to mini 2440
*
* Copyright (C) 2004 capbily - friendly-arm
* capbily@hotmail.com
*/
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdio.h>
#define BUFLEN 255

int main(int argc, char **argv)
{
      struct sockaddr_in peeraddr, / * talk each other's IP and port
storing the socket address * / localaddr; /*  Local socket address
*/
      int sockfd;
      char recmsg[BUFLEN+1];
      int socklen, n;

      if(argc!=5){
            printf("%s <dest IP address> <dest port> <source IP
address> <source port>\n",
            argv[0]);
            exit(0);
      }

      sockfd = socket(AF_INET, SOCK_DGRAM, 0);
      if(sockfd<0){
            printf("socket creating err in udptalk\n");
```

```
            exit(1);
      }
      socklen = sizeof(struct sockaddr_in);
      memset(&peeraddr, 0, socklen);
      peeraddr.sin_family=AF_INET;
      peeraddr.sin_port=htons(atoi(argv[2]));

      if(inet_pton(AF_INET, argv[1], &peeraddr.sin_addr)<=0){
            printf("Wrong dest IP address!\n");
            exit(0);
      }
      memset(&localaddr, 0, socklen);
      localaddr.sin_family=AF_INET;

      if(inet_pton(AF_INET, argv[3], &localaddr.sin_addr)<=0){
            printf("Wrong source IP address!\n");
            exit(0);
      }
      localaddr.sin_port=htons(atoi(argv[4]));

      if(bind(sockfd, &localaddr, socklen)<0){
            printf("bind local address err in udptalk!\n");
            exit(2);
      }

      if(fgets(recmsg, BUFLEN, stdin) == NULL) exit(0);
      if(sendto(sockfd, recmsg, strlen(recmsg), 0, &peeraddr,
socklen)<0){
            printf("sendto err in udptalk!\n");
            exit(3);
      }

      for(;;){
/*recv&send message loop*/
            n = recvfrom(sockfd, recmsg, BUFLEN, 0, &peeraddr,
&socklen);
            if(n<0){
                  printf("recvfrom err in udptalk!\n");
                  exit(4);
                  }
            else{
/* Successfully received packets */
                  recmsg[n]=0;
                  printf("peer:%s", recmsg);
            }
            if(fgets(recmsg, BUFLEN, stdin) == NULL) exit(0);

            if(sendto(sockfd, recmsg, strlen(recmsg), 0, &peeraddr,
socklen)<0){
                  printf("sendto err in udptalk!\n");
                  exit(3);
            }
      }
}
```

Run udptalk.c after a good compiled, /opt/FriendlyARM/mini2440/examples/, Makefile in udptalk directory create the target executable specified two files, one for the host side of the x86-udptalk, one for development board's arm-udptalk, run the make command to create these two programs together. You can use the arm-udptalk the method described above downloaded to the development board (pre-installed Linux without the program), assuming the host's IP address is 192.168.1.108, the IP address of the development board 192.168.1.230.

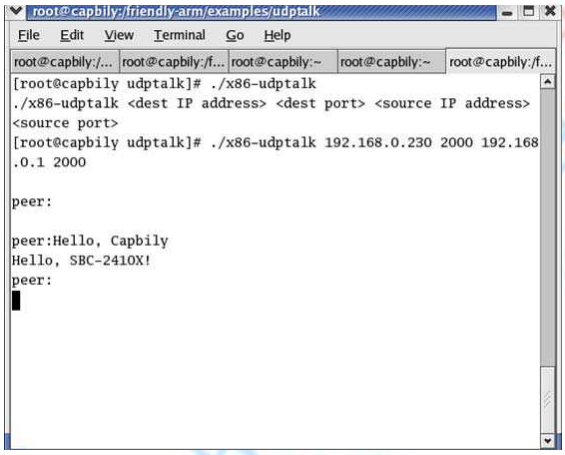
In the host terminal, enter:

```
#./x86-udptalk 192.168.1.230 2000 192.168.1.108 2000
```

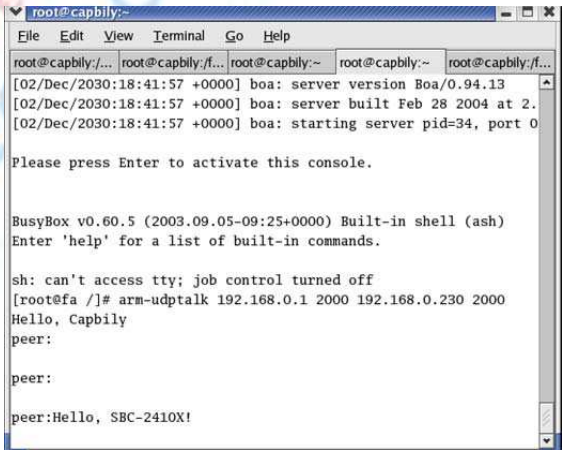
Input in the development of the terminal board.

```
#arm-udptalk 192.168.1.108 2000 192.168.1.230 2000
```

Then the results are shown below:



x86-udptalk



arm-udptalk

7.2.7 Sample math library calls

Source code description:

Test program source code directory	/opt/FriendlyARM/mini2440/examples/math
Test program source code name	mathtest.c
Test program executable file name	mathtest
Note: The key is using mathematical functions to include the header file math.h, and added at compile time math library libm.	

Source code:

```
# include <stdio.h>
# include <stdlib.h>
# include <math.h>;   //Note: Be sure to include the header file
int main (void)
{
      double a=8.733243;
      printf("sqrt(%f)=%f\n", a, sqrt(a));
      return 0;
}
```

Contents of the corresponding Makefile:

```
CROSS=arm-linuxall:
all: mathtest
# Note: there includes math library libm, red part
mathtest:
      $(CROSS)gcc -o mathtest main.c -lm
clean:
      @rm -vf mathtest *.o *~
```

You can follow the above steps to manually compile the hello program out mathtest executable file, and downloaded to the development board to run it.

7.2.8 Thread test program

Source code description:

Test program source code directory	/opt/FriendlyARM/mini2440/examples/pthread
Test program source code name	pthread_test.c
Test program executable file name	pthread_test
Note: The key is to use the thread to include the header file pthread.h, and added at compile time thread library libpthread.	

Source code:

```
#include <stddef.h>
#include <stdio.h>
#include <unistd.h>
#include "pthread.h"  ; Note: Be sure to include the header file.
void reader_function(void);
void writer_function(void);
char buffer;
int buffer_has_item=0;
pthread_mutex_t mutex;
main()
{
      pthread_t reader;
      pthread_mutex_init(&mutex,NULL);
      pthread_create(&reader,NULL,(void*)&reader_function,NULL);
      writer_function();
}

void writer_function(void)
{
      while(1)
      {
            pthread_mutex_lock(&mutex);
            if(buffer_has_item==0)
            {
                  buffer='a';
                  printf("make a new item\n");
                  buffer_has_item=1;
            }
            pthread_mutex_unlock(&mutex);
      }
}

void reader_function(void)
{
      while(1)
      {
            pthread_mutex_lock(&mutex);
```

```
            if(buffer_has_item==1)
            {
                  buffer='\0';
                  printf("consume item\n");
                  buffer_has_item=0;
            }
            pthread_mutex_unlock(&mutex);
      }
}
```

Contents of the corresponding Makefile:

```
CROSS=arm-linux
all: pthread
# Note: the department includes the thread library libphread, red
part.
pthread:
      $(CROSS)gcc -static -o pthread main.c -lpthread
clean:
      @rm -vf pthread *.o *~
```

You can follow the above steps to manually compile the hello program executable files out of pthread, and then downloaded to the development board to run it.

7.2.9 Pipeline Application Programming Examples - Web control LED

Source code description:

Test program source code directory	/opt/FriendlyARM/mini2440/examples/led-player
Test program source code name	led-player.c
Test program executable file name	led-player
Description: See description	

Web page after we can start sending commands to control development board LED blink mode, in fact, inter-process communication is a typical example of shared resources, inter-process communication is the IPC (Inter Process Communication), the general purpose inter-process communication are:

(1) Data transfer

(2) Shared data

(3) Notification event

(4) Sharing of resources

(5) Process control.

Linux supports a variety of IPC mechanisms, signals and channels is one of the two. For a more detailed inter-process communication, the general Linux programming books are introduce, we have much to say in this.

The web page to control the LED blinking pattern is achieved through the channels system, which LED is the sharing of resources, led-player is a background program, when it starts to create a named pipe /tmp/led-control (of course, the pipeline can also be created by the mknod command, so we must rewrite the program, interested can see for yourself), and has been monitoring the data input of the channels, according to different parameters (model type and cycle period) to change the LED display mode; leds.cgi is a gateway program, it receives over the character from the Web form to send commands (ping ping-pong mode or on behalf of Marquee mode, counter mode on behalf of the counter, stop on behalf of the stop mode, slow behalf cycle 0.25m, normal cycle on behalf of 0.125m, fast on behalf of cycle 0.0625m), and converts these instructions to assign the actual number, then call the echo command to the channels transmission /tmp/led-control in order to achieve control of an LED, the following is a source code of program.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <string.h>
static int led_fd;
static int type = 1;
static void push_leds(void)
{
      static unsigned step;
      unsigned led_bitmap;
      int i;
      switch(type) {
            case 0:
                  if (step >= 6) {
                        step = 0;
```

```
                }
                if (step < 3) {
                        led_bitmap = 1 << step;
                }
                else {
                        led_bitmap = 1 << (6 - step);
                }
                break;
        case 1:
                if (step > 255) {
                        step = 0;
                }
                led_bitmap = step;
                break;
        default:
                led_bitmap = 0;
    }
    step++;
    for (i = 0; i < 4; i++) {
        ioctl(led_fd, led_bitmap & 1, i);
        led_bitmap >>= 1;
    }
}
int main(void)
{
    int led_control_pipe;
    int null_writer_fd;    // for read endpoint not blocking when
control process exit
    double period = 0.5;
    led_fd = open("/dev/leds0", 0);

    if (led_fd < 0) {
        led_fd = open("/dev/leds", 0);
    }
    if (led_fd < 0) {
        perror("open device leds");
        exit(1);
    }
    unlink("/tmp/led-control");
    mkfifo("/tmp/led-control", 0666);

    led_control_pipe=open("/tmp/led-control",O_RDONLY|O_NONBLOCK);

    if (led_control_pipe < 0) {
        perror("open control pipe for read");
        exit(1);
    }
    null_writer_fd=open("/tmp/led-control",O_WRONLY|O_NONBLOCK);
    if (null_writer_fd < 0) {
        perror("open control pipe for write");
        exit(1);
    }
    for (;;) {
        fd_set rds;
        struct timeval step;
```

```
            int ret;
            FD_ZERO(&rds);
            FD_SET(led_control_pipe, &rds);
            step.tv_sec = period;
            step.tv_usec = (period - step.tv_sec) * 1000000L;
            ret = select(led_control_pipe+1,&rds,NULL,NULL,&step);
            if (ret < 0) {
                  perror("select");
                  exit(1);
            }
            if (ret == 0) {
                  push_leds();
            }
            else if (FD_ISSET(led_control_pipe, &rds)) {
                  static char buffer[200];
                  for (;;) {
                        char c;
                        int len = strlen(buffer);
                        if (len >= sizeof buffer - 1) {
                              memset(buffer, 0, sizeof buffer);
                              break;
                        }
                        if (read(led_control_pipe, &c, 1) != 1) {
                              break;
                        }
                        if (c == '\r') {
                              continue;
                        }
                        if (c == '\n') {
                              int tmp_type;
                              double tmp_period;
                              if (sscanf(buffer,"%d%lf", &tmp_type,
&tmp_period) == 2) {
                                    type = tmp_type;
                                    period = tmp_period;
                              }
                              fprintf(stderr, "type is %d, period is %lf\n",
type, period);
                              memset(buffer, 0, sizeof buffer);
                              break;
                        }
                        buffer[len] = c;
                        }
            }
      }
      close(led_fd);
      return 0;
}
```

Compiler using the make command to a led-player directly to the executable file, it is as a server placed in the development board for the /sbin directory.

Leds.cgi gateway source code (the program the development board's position: /www/leds.cgi), showing that the gateway program is actually a shell script, it is calling as a Web page leds.html implement "action":

```
#!/bin/sh
type=0
period=1
case $QUERY_STRING in
      *ping*)
            type=0
            ;;
      *counter*)
            type=1
            ;;
      *stop*)
            type=2
            ;;
esac
case $QUERY_STRING in
      *slow*)
            period=0.25
            ;;
      *normal*)
            period=0.125
            ;;
      *fast*)
            period=0.0625
            ;;
esac
/bin/echo $type $period > /tmp/led-control
echo "Content-type: text/html; charset=gb2312"
echo
/bin/cat led-result.template
exit 0
```

7.2.10 Hello world based on the C++

Source code description:

Test program source code directory	/opt/FriendlyARM/mini2440/examples/c++
Test program source code name	cplus.c++
Test program executable file name	cplus
Description: -	

Source code and notes:

```
#include <iostream>
#include <cstring>
using namespace std;
class String
{
      private:
            char *str;
      public:
            String(char *s)
            {
                  int lenght=strlen(s);
                  str = new char[lenght+1];
                  strcpy(str, s);
            }
            ~String()
            {
            cout << "Deleting str.\n";
            delete[] str;
            }
      void display()
      {
            cout << str <<endl;
      }
};
int main(void)
{
      String s1="I like FriendlyARM.";
      cout << "s1=";
      s1.display();
      return 0;
      double num, ans;

      cout << "Enter num:";
}
```

You can follow the above steps to manually compile the hello program out cplus executable file, and then downloaded to the development board to run it.