原创 zxywwb 2015-10-09 10:21:40 13651 收藏 23 版权
分类专栏： C/C++ 基于Arm板linux嵌入式系统RS485串口读写通讯 文章标签： 基于Arm板linux嵌入式系统RS48 arm linux 嵌入式系统 串口通讯

最近在做基于Arm板linux嵌入式系统的RS485串口读写通讯首先参考 http://bbs.chinaunix.net/thread-3650543-1-1.html上的文章，该文章写道，读的时候有问题，本想在该文后面做些补充，但没权限发表，只好另起炉灶，在这里接着写了

之前楼主的代码我进行了实际的调试，我将串口换成 /dev/ttySAC0（对于串口1）读基本上没问题，也是比较经典的解决方案，应该是基于IBM developworks社区的代码吧，具体也没考究，不过收的时候的确有些问题，我当时的现象是能部分收取，就是不全，多方尝试和查找网文，觉得问题应该集中在sleep时间上，应该是稍长了，可以用没有sleep的while进行read，实测接收正常，另外也还有另外两种方法接收的--信号和select，推荐select，在此把我解决问题用到的网文链接提供给诸位读者，希望能节省大家调试的时间：

[url]http://blog.csdn.net/bg2bkk/article/details/8668576[/url] （ Linux系统串口接收数据编程 ）
[url]http://blog.csdn.net/bg2bkk/article/details/8623867[/url] （ Linux串口编程 ）
[url]http://www.ibm.com/developerworks/cn/linux/l-serials/index.html[/url]（ Linux 下串口编程入门 ）
[url]http://zwkufo.blog.163.com/blog/static/258825120092171154284/[/url]（ 使用tcgetattr函数与tcsetattr函数控制终端 ）

读过如上的诸篇文章后，大家估计就会比较熟练了

转其中最有价值的一篇，大家看后就会写了：

Linux系统串口接收数据编程

http://blog.csdn.net/bg2bkk/article/details/8668576

之前基于IBM developworks社区的代码，做了串口初始化和发送的程序，今天在此基础上添加了读取串口数据的程序。首先是最简单的循环读取程序，第二个是通过软中断方式，使用信号signal机制读取串口，这里需要注意的是硬件中断是设备驱动层级的，而读写串口是用户级行为，只能通过信号机制模拟中断，信号机制的发生和处理其实于硬件中断无异，第三个是通过select系统调用，在没有数据时阻塞进程，串口有数据需要读时唤醒进程。第二个和第三个例子都能用来后台读取数据，值得学习。

代码一：循环读取数据

```cpp
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<termios.h>
#include<errno.h>

#define FALSE -1
#define TRUE 0

int speed_arr[] = { B38400, B19200, B9600, B4800, B2400, B1200, B300,B38400, B19200, B9600, B4800, B2400, B1200, B300, };
int name_arr[] = {38400,  19200,  9600,  4800,  2400,  1200,  300, 38400, 19200,  9600, 4800, 2400, 1200,  300, };
void set_speed(int fd, int speed){
  int   i;
  int   status;
  struct termios   Opt;
  tcgetattr(fd, &Opt);
  for ( i= 0;  i < sizeof(speed_arr) / sizeof(int);  i++) {
    if  (speed == name_arr[i]) {
      tcflush(fd, TCIOFLUSH);
      cfsetispeed(&Opt, speed_arr[i]);
      cfsetospeed(&Opt, speed_arr[i]);
      status = tcsetattr(fd, TCSANOW, &Opt);
      if  (status != 0) {
        perror("tcsetattr fd1");
        return;
      }
      tcflush(fd,TCIOFLUSH);
    }
  }
}

int set_Parity(int fd,int databits,int stopbits,int parity)
{
    struct termios options;
    if  ( tcgetattr( fd,&options)  !=  0) {
        perror("SetupSerial 1");
        return(FALSE);
    }
    options.c_cflag &= ~CSIZE;
    switch (databits)
    {
    case 7:
        options.c_cflag |= CS7;
        break;
    case 8:
        options.c_cflag |= CS8;
        break;
    default:
        fprintf(stderr,"Unsupported data size\n"); return (FALSE);
    }
    switch (parity)
    {
        case 'n':
        case 'N':
            options.c_cflag &= ~PARENB;   /* Clear parity enable */
            options.c_iflag &= ~INPCK;     /* Enable parity checking */
            break;
        case 'o':
        case 'O':
            options.c_cflag |= (PARODD | PARENB);
            options.c_iflag |= INPCK;             /* Disnable parity checking */
            break;
        case 'e':
        case 'E':
```

```
            options.c_cflag |= PARENB;     /* Enable parity */
            options.c_cflag &= ~PARODD;
            options.c_iflag |= INPCK;       /* Disnable parity checking */
            break;
        case 'S':
        case 's':  /*as no parity*/
            options.c_cflag &= ~PARENB;
            options.c_cflag &= ~CSTOPB;break;
        default:
            fprintf(stderr,"Unsupported parity\n");
            return (FALSE);
        }

    switch (stopbits)
    {
        case 1:
            options.c_cflag &= ~CSTOPB;
            break;
        case 2:
            options.c_cflag |= CSTOPB;
           break;
        default:
             fprintf(stderr,"Unsupported stop bits\n");
             return (FALSE);
    }
    /* Set input parity option */
    if (parity != 'n')
        options.c_iflag |= INPCK;
    tcflush(fd,TCIFLUSH);
    options.c_cc[VTIME] = 150;
    options.c_cc[VMIN] = 0; /* Update the options and do it NOW */
    if (tcsetattr(fd,TCSANOW,&options) != 0)
    {
        perror("SetupSerial 3");
        return (FALSE);
    }
    return (TRUE);
}

int main()
{
    printf("This program updates last time at %s   %s\n",__TIME__,__DATE__);
    printf("STDIO COM1\n");
    int fd;
    fd = open("/dev/ttyS0",O_RDWR);
    if(fd == -1)
    {
        perror("serialport error\n");
    }
    else
    {
        printf("open ");
        printf("%s",ttyname(fd));
        printf(" succesfully\n");
    }

    set_speed(fd,115200);
    if (set_Parity(fd,8,1,'N') == FALSE)  {
        printf("Set Parity Error\n");
        exit (0);
    }
    char buf[] = "fe55aa07bc010203040506073d";
    write(fd,&buf,26);
    char buff[512];
    int nread;
    while(1)
    {
        if((nread = read(fd, buff, 512))>0)
        {
            printf("\nLen: %d\n",nread);
            buff[nread+1] = '\0';
            printf("%s",buff);
        }
    }
    close(fd);
    return 0;
}
```

代码清单二：通过signal机制读取数据

```cpp
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/signal.h>
#include<fcntl.h>
#include<termios.h>
#include<errno.h>

#define FALSE -1
#define TRUE 0
#define flag 1
#define noflag 0

int wait_flag = noflag;
int STOP = 0;
int res;

int speed_arr[] =
  { B38400, B19200, B9600, B4800, B2400, B1200, B300, B38400, B19200, B9600,
B4800, B2400, B1200, B300, };
int name_arr[] =
  { 38400, 19200, 9600, 4800, 2400, 1200, 300, 38400, 19200, 9600, 4800, 2400,
1200, 300, };
void
set_speed (int fd, int speed)
{
  int i;
  int status;
  struct termios Opt;
```

```
  tcgetattr (fd, &Opt);
  for (i = 0; i < sizeof (speed_arr) / sizeof (int); i++)
    {
      if (speed == name_arr[i])
    {
      tcflush (fd, TCIOFLUSH);
      cfsetispeed (&Opt, speed_arr[i]);
      cfsetospeed (&Opt, speed_arr[i]);
      status = tcsetattr (fd, TCSANOW, &Opt);
      if (status != 0)
        {
          perror ("tcsetattr fd1");
          return;
        }
      tcflush (fd, TCIOFLUSH);
    }
    }
}

int
set_Parity (int fd, int databits, int stopbits, int parity)
{
  struct termios options;
  if (tcgetattr (fd, &options) != 0)
    {
      perror ("SetupSerial 1");
      return (FALSE);
    }
  options.c_cflag &= ~CSIZE;
  switch (databits)
    {
    case 7:
      options.c_cflag |= CS7;
      break;
    case 8:
      options.c_cflag |= CS8;
      break;
    default:
      fprintf (stderr, "Unsupported data size\n");
      return (FALSE);
    }
  switch (parity)
    {
    case 'n':
    case 'N':
      options.c_cflag &= ~PARENB;   /* Clear parity enable */
      options.c_iflag &= ~INPCK;    /* Enable parity checking */
      break;
    case 'o':
    case 'O':
      options.c_cflag |= (PARODD | PARENB);
      options.c_iflag |= INPCK; /* Disnable parity checking */
      break;
    case 'e':
    case 'E':
      options.c_cflag |= PARENB;    /* Enable parity */
      options.c_cflag &= ~PARODD;
      options.c_iflag |= INPCK; /* Disnable parity checking */
      break;
    case 'S':
    case 's':           /*as no parity */
      options.c_cflag &= ~PARENB;
      options.c_cflag &= ~CSTOPB;
      break;
    default:
      fprintf (stderr, "Unsupported parity\n");
      return (FALSE);
    }

  switch (stopbits)
    {
    case 1:
      options.c_cflag &= ~CSTOPB;
      break;
    case 2:
      options.c_cflag |= CSTOPB;
      break;
    default:
      fprintf (stderr, "Unsupported stop bits\n");
      return (FALSE);
    }
  /* Set input parity option */
  if (parity != 'n')
    options.c_iflag |= INPCK;
  tcflush (fd, TCIFLUSH);
  options.c_cc[VTIME] = 150;
  options.c_cc[VMIN] = 0;   /* Update the options and do it NOW */
  if (tcsetattr (fd, TCSANOW, &options) != 0)
    {
      perror ("SetupSerial 3");
      return (FALSE);
    }
  return (TRUE);
}

void
signal_handler_IO (int status)
{
  printf ("received SIGIO signale.\n");
  wait_flag = noflag;
}

int
main ()
{
  printf ("This program updates last time at %s   %s\n", __TIME__, __DATE__);
  printf ("STDIO COM1\n");
  int fd;
  struct sigaction saio;
  fd = open ("/dev/ttyUSB0", O_RDWR);
  if (fd == -1)
    {
      perror ("serialport error\n");
```

```
    }
  else
    {
      printf ("open ");
      printf ("%s", ttyname (fd));
      printf (" succesfully\n");
    }

  saio.sa_handler = signal_handler_IO;
  sigemptyset (&saio.sa_mask);
  saio.sa_flags = 0;
  saio.sa_restorer = NULL;
  sigaction (SIGIO, &saio, NULL);

  //allow the process to receive SIGIO
  fcntl (fd, F_SETOWN, getpid ());
  //make the file descriptor asynchronous
  fcntl (fd, F_SETFL, FASYNC);

  set_speed (fd, 115200);
  if (set_Parity (fd, 8, 1, 'N') == FALSE)
    {
      printf ("Set Parity Error\n");
      exit (0);
    }

  char buf[255];
while (STOP == 0)
    {
      usleep (100000);
      /* after receving SIGIO ,wait_flag = FALSE,input is availabe and can be read */
      if (wait_flag == 0)
    {
      memset (buf, 0, sizeof(buf));
      res = read (fd, buf, 255);
      printf ("nread=%d,%s\n", res, buf);
//    if (res ==1)
//      STOP = 1;       /*stop loop if only a CR was input */
      wait_flag = flag; /*wait for new input */
    }
    }


  close (fd);
  return 0;
}
```

代码三：通过select系统调用进行io多路切换，实现异步读取串口数据

```python
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/signal.h>
#include<fcntl.h>
#include<termios.h>
#include<errno.h>

#define FALSE -1
#define TRUE 0
#define flag 1
#define noflag 0

int wait_flag = noflag;
int STOP = 0;
int res;

int speed_arr[] =
  { B38400, B19200, B9600, B4800, B2400, B1200, B300, B38400, B19200, B9600,
B4800, B2400, B1200, B300, };
int name_arr[] =
  { 38400, 19200, 9600, 4800, 2400, 1200, 300, 38400, 19200, 9600, 4800, 2400,
1200, 300, };
void
set_speed (int fd, int speed)
{
  int i;
  int status;
  struct termios Opt;
  tcgetattr (fd, &Opt);
  for (i = 0; i < sizeof (speed_arr) / sizeof (int); i++)
    {
      if (speed == name_arr[i])
    {
      tcflush (fd, TCIOFLUSH);
      cfsetispeed (&Opt, speed_arr[i]);
      cfsetospeed (&Opt, speed_arr[i]);
      status = tcsetattr (fd, TCSANOW, &Opt);
      if (status != 0)
        {
          perror ("tcsetattr fd1");
          return;
        }
      tcflush (fd, TCIOFLUSH);
    }
    }
}

int
set_Parity (int fd, int databits, int stopbits, int parity)
{
  struct termios options;
  if (tcgetattr (fd, &options) != 0)
    {
      perror ("SetupSerial 1");
      return (FALSE);
    }
  options.c_cflag &= ~CSIZE;
  switch (databits)
```

```
    {
    case 7:
      options.c_cflag |= CS7;
      break;
    case 8:
      options.c_cflag |= CS8;
      break;
    default:
      fprintf (stderr, "Unsupported data size\n");
      return (FALSE);
    }
  switch (parity)
    {
    case 'n':
    case 'N':
      options.c_cflag &= ~PARENB;   /* Clear parity enable */
      options.c_iflag &= ~INPCK;    /* Enable parity checking */
      break;
    case 'o':
    case 'O':
      options.c_cflag |= (PARODD | PARENB);
      options.c_iflag |= INPCK; /* Disnable parity checking */
      break;
    case 'e':
    case 'E':
      options.c_cflag |= PARENB;    /* Enable parity */
      options.c_cflag &= ~PARODD;
      options.c_iflag |= INPCK; /* Disnable parity checking */
      break;
    case 'S':
    case 's':           /*as no parity */
      options.c_cflag &= ~PARENB;
      options.c_cflag &= ~CSTOPB;
      break;
    default:
      fprintf (stderr, "Unsupported parity\n");
      return (FALSE);
    }

  switch (stopbits)
    {
    case 1:
      options.c_cflag &= ~CSTOPB;
      break;
    case 2:
      options.c_cflag |= CSTOPB;
      break;
    default:
      fprintf (stderr, "Unsupported stop bits\n");
      return (FALSE);
    }
  /* Set input parity option */
  if (parity != 'n')
    options.c_iflag |= INPCK;
  tcflush (fd, TCIFLUSH);
  options.c_cc[VTIME] = 150;
  options.c_cc[VMIN] = 0;   /* Update the options and do it NOW */
  if (tcsetattr (fd, TCSANOW, &options) != 0)
    {
      perror ("SetupSerial 3");
      return (FALSE);
    }
  return (TRUE);
}

void
signal_handler_IO (int status)
{
  printf ("received SIGIO signale.\n");
  wait_flag = noflag;
}

int
main ()
{
  printf ("This program updates last time at %s   %s\n", __TIME__, __DATE__);
  printf ("STDIO COM1\n");
  int fd;
  fd = open ("/dev/ttyUSB0", O_RDWR);
  if (fd == -1)
    {
      perror ("serialport error\n");
    }
  else
    {
      printf ("open ");
      printf ("%s", ttyname (fd));
      printf (" succesfully\n");
    }

  set_speed (fd, 115200);
  if (set_Parity (fd, 8, 1, 'N') == FALSE)
    {
      printf ("Set Parity Error\n");
      exit (0);
    }

  char buf[255];
  fd_set rd;
  int nread = 0;
  while(1)
  {
    FD_ZERO(&rd);
    FD_SET(fd, &rd);
    while(FD_ISSET(fd, &rd))
    {
        if(select(fd+1, &rd, NULL,NULL,NULL) < 0)
        {
            perror("select error\n");
        }
        else
        {
            while((nread = read(fd, buf, sizeof(buf))) > 0)
```

```
            {
                printf("nread = %d,%s\n",nread, buf);
                printf("test\n");
                memset(buf, 0 , sizeof(buf));
            }
        }
    }
  }
  close (fd);
  return 0;
}
```

RS485通信基于Linux下 11-13
这是基于Linux下采用RS485通信基础。代码的详解，没部分都有，。

arm-linux详细的串口调试（适合初学者） 04-27
适合初学者，对于arm-linux的串口进行掌握，很方便的从超级终端和串口工具看到数据的发送与接收。

优质评论可以帮助作者获得更高权重 评论

天泉证道： 能收到消息吗，请教一下问题 1 年前 回复

遥_望： select本质来讲，也是同步IO的一种 2 年前 回复

rs485对应的linux串行终端,基于ARM9与LINUX的RS485总线... 8-14
RS485驱动程序修改完成后,可以像操作串口一样操作RS485接口。在嵌入式Linux系统下,串口的设备文件位于/dev目录下,可以使用文件打开、读写函数[2,...

linux嵌入式 控制io口,基于ARM9嵌入式的RS485总线接口... 8-3
设计使用ARM9处理器S3C2440内部集成的UART外设和RSM485模块构建具有电源隔离、电气隔离、总线保护的RS485总线接口,通过对嵌入式Linux系统...

VB485通信串口通信实例 06-02
VB串口通信实例 含源代码，有与仪表 PLC 单片机，5166卡通信，都含源码，希望对大家有用

RS485串口测试软件 03-26
该软件可以测试RS485串口通讯，已经试验过，好用！

arm linux uart配置成rs485模式,基于ARM9与LINUX的RS48... 8-5
摘要:在ARM9处理器S3C2440上设计RS485通信接口,实现与总线上其他设备的通信,利用ARM9处理器内部集成的UART外设和RSM485CT模块组成RS485...

RS485通信(Modbus)丢包解决经验(基于ARM/Linux和STM32... 8-19
1. RS485通信属于半双工通信,即发送和接收不能同时进行,需要切换,一般定义一根GPIO做RTS切换。2. ARM-Linux平台有Linux底层驱动做自动切换,比如...

rs485通讯（C++） 04-05
PC机和RS485串口通讯程序，可以灵活修改，功能相当于串口调试助手的精简版

基于arm的串口读取以及存储 03-01
该代码给出了基于arm硬件基础的串口数据读取，以及将数据写入sd卡

利用开发板上的RS485模块与PC机的串口通讯_hanpengyu的... 8-7
然后再利用单片机的一个引脚作为DIR(RS485是半双工的,通过dir实现发送和接受的控制) 二、将智能仪表RS485接口转换 将智能仪表的B-,A+与PL2303芯...

迅为IMX6ULL开发板Linux RS232/485驱动实验（下） mucheni的博客 177
53.3硬件原理图 53.3.1 RS232 UART3 通过跳线帽 J46 来设置成 TTL 电平的 RS232 信号。连接 J46 的 3-5 和 4-6 后，UART3 通过 TTL 电平输出。 53....

Linux RS232/485/GPS 驱动实验(3)-设备树修改 qq_45761493的博客 15
1、UART3 IO 节点创建 UART3 用到了 UART3_TXD 和 UART3_RXD 这两个 IO，因此要先在 iomuxc 中创建 UART3 对应的 pinctrl 子节点，在 iomuxc 中...

linux串口配置成485,am335x 如何使用将串口配置成485功能 weixin_35779001的博客 60
内核版本是linux-rt-4.1.18，现在想修改uart2，使用rs485的功能，使用gpio0_5作为控制引脚配置完成没有办法使用，需要手动对gpio0_5拉高或拉低才能...

正点原子j-link驱动下载_正点原子Linux第六十三章Linux RS232/485/GPS驱动实验 weixin_39578513的博客 165
1)资料下载:点击资料即可下载2)对正点原子Linux感兴趣的同学可以加群讨论：9354467413 ）关注正点原子公众号，获取最新资料更新第六十三章Linux R...

Linux RS485 通信代码 03-20
本资源给出了Linux下进行RS485通信的实例代码

linux下rs485驱动 11-21
linux下rs485驱动，可支持115200波特率的数据的收发

基于RS485的ModBus通讯 Amanda1m的博客 1756
一、通信 1.信道 传输数据的载体，比如古代传信息白天用狼烟，晚上用烽火。 2.协议 收发数据双方达成的一致协议，对数据格式、含义等作出规定，比...

终于讲透了，史上最详细的RS485串口通讯！ 嵌入式Linux 3万+
RS485接口组成的半双工网络，一般是两线制，多采用屏蔽双绞线传输，这种接线方式为总线式拓扑结构在同一总线上最多可以挂接32个结点。我们知道...

linux rs 485应用程序,rs485驱动设计 - 基于ARM9与LINUX的RS485总线的通信接口设计 weixin_42122878的博客 18
2.2RS485驱动设计设计中使用ARM9处理器S3C2440内部集成的UART外设和RSM485模块构建而成，其驱动程序与RS232驱动程序相比多了一个通信方...

IMX6 RS485 驱动移植 你学习了吗？咋办，先想法子忽悠成'砖家'呗 2193
1、在内核源码根目录应用附件的补丁： patch -p1 < 6g2c-rs485.patch 2、修改设备树中对应串口设备节点属性，以uart4、使能引脚gpio1_14为例： &u...

DFI协议3.0~5.0.rar 09-09
DFI协议：DDR_PHY_Interface_Specification__v3_0、DDR_PHY_Interface_Specification__v4_0、DDR_PHY_Interface_Specification__v3_1、DDR_P...

《中国医疗器械杂志》"监管与测试"栏目入选华东地区期刊"优秀栏目".pdf 09-09
《中国医疗器械杂志》"监管与测试"栏目入选华东地区期刊"优秀栏目".pdf

行业-电子政务-动态省电模式及基于业务型式的配置改变.zip 09-09
行业-电子政务-动态省电模式及基于业务型式的配置改变.zip

行业-电子政务-电子元件装置的电气特性测定方法、电子元件装置的筛选方法及电气特性测定装置.zip 09-09
行业-电子政务-电子元件装置的电气特性测定方法、电子元件装置的筛选方法及电气特性测定装置.zip

zxywwb
码龄17年 暂无认证

5	25万+	167万+	5万+	
原创	周排名	总排名	访问	等级

828	9	6	8	24
积分	粉丝	获赞	评论	收藏

私信 关注

搜博主文章

热门文章

基于Arm板linux嵌入式系统RS485串口读写通讯 13647

用VB6自带打包工具创建桌面快捷方式和卸载程序 5290

网摘：行走社会的100条忠告 3706

linux 远程调用命令 2865

[X86] 关于/usr下的ccs，df，sfw，ucb，openwin，xpg4，xpg6的含义 2221

最新评论

基于Arm板linux嵌入式系统RS485串口读...
天泉证道: 能收到消息吗，请教一下问题_

基于Arm板linux嵌入式系统RS485串口读...
遥_望: select本质来讲，也是同步IO的一...

用VB6自带打包工具创建桌面快捷方式和...
lovenets2008: ""由于字数限制，接上： Dim strLinkName As String strLinkName =...

用VB6自带打包工具创建桌面快捷方式和...
lovenets2008: '思路很好，但是上面有BUG（ InStrRev(strDeskTopPath, "/")和"$(Pr...

用VB6自带打包工具创建桌面快捷方式和...
马上有运气: 不起作用啊！

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

[X86] 关于/usr下的ccs，df，sfw，ucb，openwin，xpg4，xpg6的含义

用VB6自带打包工具创建桌面快捷方式和卸载程序

linux 远程调用命令

2015年 1篇	2011年 1篇
2009年 15篇	2008年 5篇