第三讲 嵌入式Linux串口程序设计

华中科技大学电信学院
鄢舒
E-mail: yan0shu@gmail.com

内容提纲(1/4)

- 1. 嵌入式Linux串口编程概述
- 2. 嵌入式Linux串口详细配置
- 3. 嵌入式Linux串口编程示例
- 4. 实验内容与要求

1.1 数据通信的基本方式

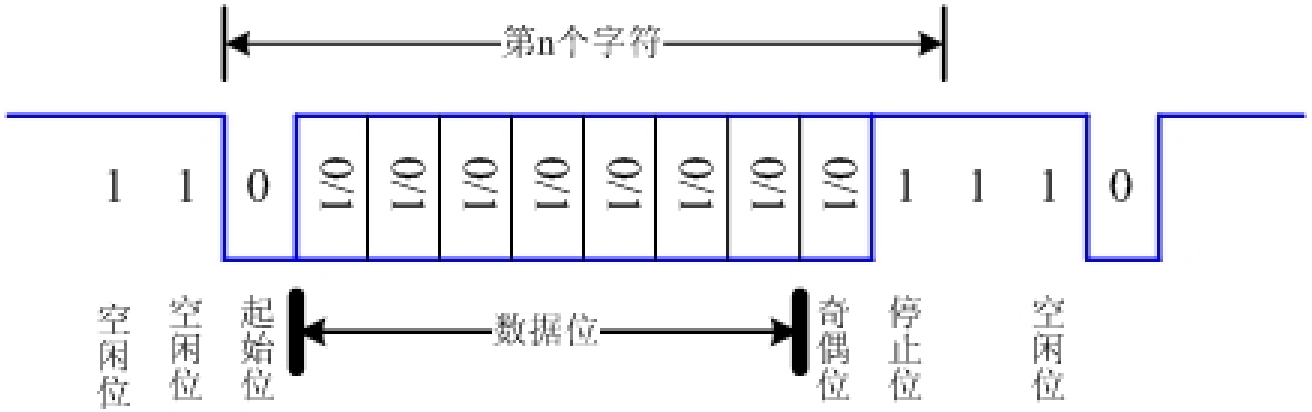
用户常见的数据通信的基本方式有两种：

- 并行通信；
- 串行通信；

- 串行通信是计算机常用的接口，如：RS-232-C接口。该标准规定采用一个DB25芯引脚连接器或DB9芯引脚连接器。
- 芯片内部常具有UART控制器，其可工作于Interrupt(中断模式)或DMA（直接内存访问）模式。

1.2 串行通信原理

- 采用异步串行I/O方式；
- 将传输数据的每一个字符一位接一位地传送；
- 各个不同位分时地使用同一传输通道；
- 开始时，线路处于空闲状态，送出连续“1”；
- 传送开始时，首先发送一个“0”作为起始位，然后出现在通信线上的是字符的二进制编码数据，每个字符的数据位长度可以为5、6、7位或8位，一般采用ASCII编码。后面是奇偶校验位。最后是表示停止位的“1”信号，这个停止位可持续1、1.5、2位的时间宽度。
- 每个数据位的宽度等于波特率的倒数。



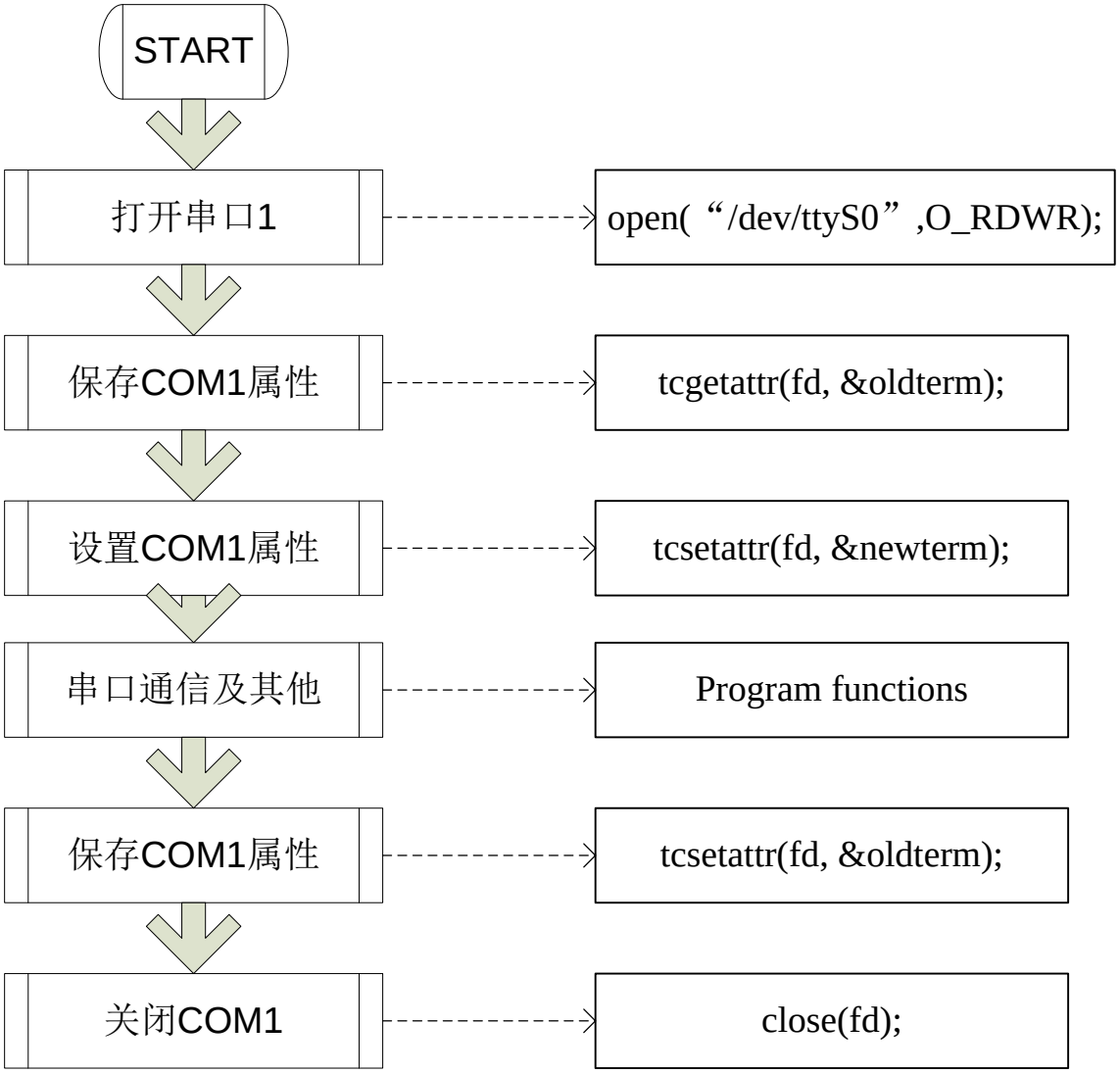
1.3 UART的操作

- UART的操作主要包括以下几个部分：
  - 数据发送；
  - 数据接收；
  - 产生中断；
  - 产生波特率；
  - Loopback模式；
  - 红外模式；
  - 自动流控模式；
- 串口参数的配置主要包括：波特率、数据位、停止位、流控协议。

1.4 Linux中的串口设备

- Linux中的串口设备文件存放于/dev目录下，其中串口一(COM1)，串口二(COM2)对应设备名依次为“/dev/ttyS0”、“/dev/ttyS1”。
- 在Linux下操作串口与操作文件相同。

1.5 串口通信的流程


START
打开串口1
open(“/dev/ttyS0”,O_RDWR);
保存COM1属性
tcgetattr(fd, &oldterm);
设置COM1属性
tcsetattr(fd, &newterm);
串口通信及其他
Program functions
保存COM1属性
tcsetattr(fd, &oldterm);
关闭COM1
close(fd);

内容提纲(2/4)

- 1. 嵌入式Linux串口编程概述
- 2. 嵌入式Linux串口详细配置
- 3. 嵌入式Linux串口编程示例
- 4. 实验内容与要求

2.1 串口设置结构

- 在使用串口之前必须设置相关配置，包括：波特率、数据位、校验位、停止位等。串口设置由下面结构体实现：

```
struct termios {
  tcflag_t  c_iflag;     /* input flags */
  tcflag_t  c_oflag;     /* output flags */
  tcflag_t  c_cflag;     /* control flags */
  tcflag_t  c_lflag;     /* local flags */
  cc_t      c_cc[NCCS]; /* control characters */
};
```

2.2 c_cflag成员变量

- termios结构中c_cflag最为重要，可设置波特率、数据位、校验位、停止位。在设置波特率时需在数字前加上‘B’，如B9600。B19200。使用其需通过“与”“或”操作方式

c_cflag	CCTS_OFLOW	输出的CTS流控制
	CIGNORE	忽略控制标志
	CLOCAL	忽略解制-解调器状态行
	CREAD	启用接收装置
	CRTS_IFLOW	输入的RTS流控制
	CSIZE	字符大小屏蔽
	CSTOPB	送两个停止位，否则为1位
	HUPCL	最后关闭时断开
	MDMBUF	经载波的流控输出
	PARENB	进行奇偶校
	PARODD	奇校，否则为偶校

2.3 c_iflag成员变量

- 输入模式c_iflag成员控制端口接收端的字符输入处理。

c_iflag	BRKINT	接到BREAK时产生SIGINT
	ICRNL	将输入的CR转换为NL
	IGNBRK	忽略BREAK条件
	IGNCR	忽略CR
	IGNPAR	忽略奇偶错字符
	IMAXBEL	在输入队列空时振铃
	INLCR	将输入的NL转换为CR
	INPCK	打开输入奇偶校验
	ISTRIP	剥除输入字符的第8位
	IUCLC	将输入的大写字符转换成小写字符
	IXANY	使任一字符都重新起动输出
	IXOFF	使起动/停止输入控制流起作用
	IXON	使起动/停止输出控制流起作用
	PARMRK	标记奇偶错

2.4 串口控制函数

tcgetattr	取属性(termios结构)
tcsetattr	设置属性(termios结构)
cfgetispeed	得到输入速度
cfgetospeed	得到输出速度
cfsetispeed	设置输入速度
cfsetospeed	设置输出速度
tcdrain	等待所有输出都被传输
tcflow	挂起传输或接收
tcflush	刷清未决输入和/或输出
tcsendbreak	送BREAK字符
tcgetpgrp	得到前台进程组ID
tcsetpgrp	设置前台进程组ID

2.5 串口配置流程(1/4)

1. 保存原先串口配置使用tcgetattr(fd,&oldtio)函数

   struct termios newtio,oldtio;

   tcgetattr(fd,&oldtio);

2. 激活选项有CLOCAL和CREAD,用于本地连接和接收使能。

   newtio.c_cflag | = CLOCAL | CREAD;

3. 设置波特率，使用函数cfsetispeed、 cfsetospeed

   cfsetispeed(&newtio, B115200);

   cfsetospeed(&newtio, B115200);

4. 设置数据位，需使用掩码设置。

   newtio.c_cflag &= ~CSIZE;

   newtio.c_cflag |= CS8;

2.5 串口配置流程(2/4)

5. 设置奇偶校验位，使用c_cflag和c_iflag。

设置奇校验：

```
newtio.c_cflag |= PARENB;
newtio.c_cflag |= PARODD;
newtio.c_iflag |= (INPCK | ISTRIP);
```

设置偶校验：

```
newtio.c_iflag |= (INPCK | ISTRIP);
newtio.c_cflag |= PARENB;
newtio.c_cflag &= ~PARODD;
```

6. 设置停止位，通过激活c_cflag中的CSTOPB实现。若停止位为1，则清除CSTOPB，若停止位为2，则激活CSTOPB。

```
newtio.c_cflag &= ~CSTOPB;
```

2.5 串口配置流程(3/4)

7. 设置最少字符和等待时间，对于接收字符和等待时间没有特别要求时，可设为0。

```
newtio.c_cc[VTIME]  = 0;
newtio.c_cc[VMIN] = 0;
```

8. 处理要写入的引用对象

tcflush函数刷清（抛弃）输入缓存（终端驱动程序已接收到，但用户程序尚未读）或输出缓存（用户程序已经写，但尚未发送）。

```
int tcflush (int filedes, int queue)
```

queue数应当是下列三个常数之一：

- TCIFLUSH 刷清输入队列。
- TCOFLUSH 刷清输出队列。
- TCIOFLUSH 刷清输入、输出队列。

如：tcflush(fd,TCIFLUSH);

2.5 串口配置流程(4/4)

9. 激活配置。在完成配置后，需激活配置使其生效。使用tsettattr()函数。原型：

int tcgetattr(int filedes, struct termios *termptr);

int tcsetattr(int filedes, int opt, const struct termios * termptr);

tcsetattr的参数opt使我们可以指定在什么时候新的终端属性才起作用。opt可以指定为下列常数中的一个：

- TCSANOW 更改立即发生。
- TCSADRAIN 发送了所有输出后更改才发生。若更改输出参数则应使用此选择项。
- TCSAFLUSH 发送了所有输出后更改才发生。更进一步，在更改发生时未读的所有输入数据都被删除(刷清)。

使用如：tcsetattr(fd,TCSANOW,&newtio)

内容提纲(3/4)

- 1. 嵌入式Linux串口编程概述
- 2. 嵌入式Linux串口详细配置
- 3. 嵌入式Linux串口编程示例
- 4. 实验内容与要求

3.1 Linux中使用串口的基本方法

- 在配置完串口的相关属性后，就可对串口进行打开，读写操作了。其使用方式与文件操作一样，区别在于串口是一个终端设备。

3.2 打开串口操作

```
fd = open( "/dev/ttyS0",
        O_RDWR|O_NOCTTY|O_NDELAY);
```

open函数中除普通参数外，另有两个参数O_NOCTTY和O_NDELAY。

- O_NOCTTY：通知Linux系统，这个程序不会成为这个端口的控制终端。
- O_NDELAY：通知Linux系统不关心DCD信号线所处的状态（端口的另一端是否激活或者停止）。

3.3 串口操作的步骤

- 首先打开串口；
- 然后，恢复串口的状态为阻塞状态，用于等待串口数据的读入。用fcntl函数：

  fcntl（fd, F_SETFL, 0）；
- 接着，测试打开的文件描述府是否引用一个终端设备，以进一步确认串口是否正确打开。

  isatty(STDIN_FILENO);

3.4 读写串口操作

- 串口的读写与普通文件一样，使用read,write函数。

  read(fd,buff,8);

  write(fd,buff,8);

3.5 串口操作的实例(1/2)

```
#include <stdio.h>  #include <string.h> #include <sys/types.h> #include <errno.h> #include <sys/stat.h>
#include <fcntl.h> #include <unistd.h> #include <termios.h> #include <stdlib.h>
int set_opt(int fd,int nSpeed, int nBits, char nEvent, int nStop)
{
        struct termios newtio,oldtio;
        if  ( tcgetattr( fd,&oldtio)  !=  0) {  perror("SetupSerial 1"); return -1; }
        bzero( &newtio, sizeof( newtio ) );
        newtio.c_cflag  |=  CLOCAL | CREAD;  newtio.c_cflag &= ~CSIZE;
        switch( nBits ){
        case 7: newtio.c_cflag |= CS7; break;
        case 8: newtio.c_cflag |= CS8; break;}
        switch( nEvent ){
        case 'O': newtio.c_cflag |= PARENB; newtio.c_cflag |= PARODD;
                  newtio.c_iflag |= (INPCK | ISTRIP); break;
        case 'E':  newtio.c_iflag |= (INPCK | ISTRIP); newtio.c_cflag |= PARENB;
                   newtio.c_cflag &= ~PARODD; break;
        case 'N':  newtio.c_cflag &= ~PARENB; break;}
        switch( nSpeed ){
        case 2400: cfsetispeed(&newtio, B2400); cfsetospeed(&newtio, B2400); break;
        case 4800: cfsetispeed(&newtio, B4800); cfsetospeed(&newtio, B4800); break;
        case 9600: cfsetispeed(&newtio, B9600); cfsetospeed(&newtio, B9600); break;
        case 115200: cfsetispeed(&newtio, B115200); cfsetospeed(&newtio, B115200); break;
        default: cfsetispeed(&newtio, B9600); cfsetospeed(&newtio, B9600); break;}
        if( nStop == 1 )  newtio.c_cflag &=  ~CSTOPB;
        else if ( nStop == 2 ) newtio.c_cflag |=  CSTOPB;
        newtio.c_cc[VTIME]  = 0; newtio.c_cc[VMIN] = 0; tcflush(fd,TCIFLUSH);
        if((tcsetattr(fd,TCSANOW,&newtio))!=0){ perror("com set error");return -1;}
        printf("set done!\n");return 0;}
```

3.5 串口操作的实例(2/2)

```
int open_port(int fd,int comport)
{
    if (comport==1){ fd = open( "/dev/ttyS0", O_RDWR|O_NOCTTY|O_NDELAY);
                    if (-1 == fd){ perror("Can't Open Serial Port");return -1; }
                   else printf("open ttyS0 .....\n");}
    else if(comport==2){ fd = open( "/dev/ttyS1", O_RDWR|O_NOCTTY|O_NDELAY);
                        if (-1 == fd){ perror("Can't Open Serial Port"); return -1;}
                        else printf("open ttyS1 .....\n");}
    else if (comport==3){ fd = open( "/dev/ttyS2", O_RDWR|O_NOCTTY|O_NDELAY);
                         if (-1 == fd){ perror("Can't Open Serial Port");return -1;}
                         else printf("open ttyS2 .....\n");}
    if(fcntl(fd, F_SETFL, 0)<0) printf("fcntl failed!\n");
    else printf("fcntl=%d\n",fcntl(fd, F_SETFL,0));
    if(isatty(STDIN_FILENO)==0) printf("standard input is not a terminal device\n");
    else printf("isatty success!\n");
    printf("fd-open=%d\n",fd); return fd;}
int main(void)
{   int fd;int nread,i; char buff[]="Hello  \n";
    if((fd=open_port(fd,1))<0){perror("open_port error");return -1;}
    if((i=set_opt(fd,115200,8,'N',1))<0){perror("set_opt error");return -1;}
    printf("fd=%d\n",fd); write(fd,buff,8); nread=read(fd,buff,8);
    printf("nread=%d,%s\n",nread,buff); close(fd); return 0;
}
```

内容提纲(4/4)

- 1. 嵌入式Linux串口编程概述
- 2. 嵌入式Linux串口详细配置
- 3. 嵌入式Linux串口编程示例
- 4. 实验内容与要求

4.1 本次实验要求

- 阅读理解参考程序的源码；
- 在参考程序的基础上实现如下功能：
  - 能够通过串口实现PC机和实验平台之间的简单聊天功能：运行所编写的程序后，在宿主机和目标机上的终端窗口（可以telnet到目标机上）上输入若干个单词，回车后能够在对方的终端窗口上显示出来。
- 编译应用程序；
- 下载、调试、运行。

4.2 实验注意事项

- 在启动实验平台并设置好ip后，可以使用telnet登录实验平台，以便空出串口ttyS0；
- 在实验平台上的串口设备包括：
  - COM1： /dev/ttyS0
  - COM2： /dev/ttyS1
- 使用Minicom登录过的串口(串口1)来进行实验可能会出一些问题，故可以使用另一个串口(串口2)来进行串口通信的实验。