转载 believe209 2014-06-20 16:37:24 18704 收藏 12
分类专栏： linux 文章标签： 串口通信 usb

此文章主要以串口为例子进行讲解，USB只需要修改打开的终端设备就可以

串口简介

串行口是计算机一种常用的接口，具有连接线少，通讯简单，得到广泛的使用。常用的串口是 RS-232-C 接口（又称 EIA RS-232-C）它是在1970 年由美国电子工业协会（EIA）联合贝尔系统、调制解调器厂家及计算机终端生产厂家共同制定的用于串行通讯的标准。它的全名是"数据终端设备（DTE）和数据通讯设备（DCE）之间串行二进制数据交换接口技术标准"该标准规定采用一个 25 个脚的 DB25 连接器，对连接器的每个引脚的信号内容加以规定，还对各种信号的电平加以规定。传输距离在码元畸变小于 4% 的情况下，传输电缆长度应为 50 英尺。

Linux 操作系统从一开始就对串行口提供了很好的支持，本文就 Linux 下的串行口通讯编程进行简单的介绍，如果要非常深入了解，建议看看本文所参考的《Serial Programming Guide for POSIX Operating Systems》

计算机串口的引脚说明

序号	信号名称	符号	流向	功能
3	发送数据	TXD	DTE→DCE	DTE发送串行数据
2	接收数据	RXD	DTE←DCE	DTE 接收串行数据
7	请求发送	RTS	DTE→DCE	DTE 请求 DCE 将线路切换到发送方式
8	允许发送	CTS	DTE←DCE	DCE 告诉 DTE 线路已接通可以发送数据
6	数据设备准备好	DSR	DTE←DCE	DCE 准备好
5	信号地	SG		信号公共地
1	载波检测	DCD	DTE←DCE	表示 DCE 接收到远程载波
4	数据终端准备好	DTR	DTE→DCE	DTE 准备好
9	振铃指示	RI	DTE←DCE	表示 DCE 与线路接通，出现振铃

串口操作

串口操作需要的头文件

```cpp
#include     <stdio.h>      /*标准输入输出定义*/
#include     <stdlib.h>     /*标准函数库定义*/
#include     <unistd.h>     /*Unix 标准函数定义*/
#include     <sys/types.h>
#include     <sys/stat.h>
#include     <fcntl.h>      /*文件控制定义*/
#include     <termios.h>    /*PPSIX 终端控制定义*/
#include     <errno.h>      /*错误号定义*/
```

打开串口（USB）

在 Linux 下串口文件是位于 /dev 下的

串口一 为 /dev/ttyS0（/dev/ttyUSB0）

串口二 为 /dev/ttyS1(/dev/ttyUSB1)

打开串口是通过使用标准的文件打开函数操作：

```
int fd;
/*以读写方式打开串口*/
fd = open( "/dev/ttyS0", O_RDWR);
if (-1 == fd){
/* 不能打开串口一*/
perror(" 提示错误！");
}
```

终端设备设置

最基本的设置包括波特率设置，效验位和停止位设置。

串口的设置主要是设置 struct termios 结构体的各成员值。

```
struct termio
{ unsigned short  c_iflag; /* 输入模式标志 */
  unsigned short  c_oflag;   /* 输出模式标志 */
  unsigned short  c_cflag;   /* 控制模式标志*/
  unsigned short  c_lflag;   /* local mode flags */
  unsigned char   c_line;       /* line discipline */
  unsigned char   c_cc[NCC];    /* control characters */
};
```

其具体意义如下。
c_iflag：输入模式标志，控制终端输入方式，具体参数如表1所示。
表1 c_iflag参数表

键 值	说 明
IGNBRK	忽略BREAK键输入
BRKINT	如果设置了IGNBRK，BREAK键输入将被忽略
IGNPAR	忽略奇偶校验错误
PARMRK	标识奇偶校验错误
INPCK	允许输入奇偶校验
ISTRIP	去除字符的第8个比特
INLCR	将输入的NL（换行）转换成CR（回车）
IGNCR	忽略输入的回车
ICRNL	将输入的回车转化成换行（如果IGNCR未设置的情况下）
IUCLC	将输入的大写字符转换成小写字符（非POSIX）
IXON	允许输入时对XON/XOFF流进行控制
IXANY	输入任何字符将重启停止的输出
IXOFF	允许输入时对XON/XOFF流进行控制

分类专栏
gdb 1篇
C#之winform基础 2篇
C 7篇
C++ 5篇
Data Structure 6篇
network 2篇
protocol
unix 9篇
linux 54篇
VC 3篇
趣味题
QT 12篇
嵌入式 12篇
VMWare 3篇
java 8篇
android 17篇
ubuntu 9篇
协议 6篇
驱动 5篇
标准 1篇
硬件 3篇
串口 6篇
USB 4篇
java
JNI 8篇
gstreamer 9篇
线程 3篇
进程 1篇
uboot 3篇
C# 69篇
fedora 2篇
windows 17篇
VNC 13篇
GPS
GPRS
centos 1篇
linux命令 6篇
IOS 12篇
bluetooth 21篇
blueZ 3篇
Vxworks
工具 1篇
BlueDroid 4篇
SQL Server 21篇

IMAXBEL　　当输入队列满的时候开始响铃

c_oflag：输出模式标志，控制终端输出方式，具体参数如表2所示。
　表2 c_oflag参数

键 值　说 明
OPOST　处理后输出
OLCUC　将输入的小写字符转换成大写字符（非POSIX）
ONLCR　将输入的NL（换行）转换成CR（回车）及NL（换行）
OCRNL　将输入的CR（回车）转换成NL（换行）
ONOCR　第一行不输出回车符
ONLRET　不输出回车符
OFILL　　发送填充字符以延迟终端输出
OFDEL　以ASCII码的DEL作为填充字符，如果未设置该参数，填充字符为NUL
NLDLY　换行输出延时，可以取NL0（不延迟）或NL1（延迟0.1s）
CRDLY　回车延迟，取值范围为：CR0、CR1、CR2和 CR3
TABDLY　水平制表符输出延迟，取值范围为：TAB0、TAB1、TAB2和TAB3
BSDLY　空格输出延迟，可以取BS0或BS1
VTDLY　垂直制表符输出延迟，可以取VT0或VT1
FFDLY　换页延迟，可以取FF0或FF1

c_cflag：控制模式标志，指定终端硬件控制信息，具体参数如表3所示。
　表3 c_cflag参数

键 值　　说 明
CBAUD　　波特率（4+1位）（非POSIX）
CBAUDEX　　附加波特率（1位）（非POSIX）
CSIZE　　字符长度，取值范围为CS5、CS6、CS7或CS8
CSTOPB　　设置两个停止位
CREAD　　使用接收器
PARENB　　使用奇偶校验
PARODD　　对输入使用奇偶校验，对输出使用偶校验
HUPCL　　关闭设备时挂起
CLOCAL　　忽略调制解调器线路状态
CRTSCTS　　使用RTS/CTS流控制

c_lflag：本地模式标志，控制终端编辑功能，具体参数如表4所示。
　表4 c_lflag参数

键 值　说 明
ISIG　　当输入INTR、QUIT、SUSP或DSUSP时，产生相应的信号
ICANON　　使用标准输入模式
XCASE　　在ICANON和XCASE同时设置的情况下，终端只使用大写。
ECHO　　显示输入字符
ECHOE　　如果ICANON同时设置，ERASE将删除输入的字符
ECHOK　　如果ICANON同时设置，KILL将删除当前行
ECHONL　　如果ICANON同时设置，即使ECHO没有设置依然显示换行符
ECHOPRT　如果ECHO和ICANON同时设置，将删除打印出的字符（非POSIX）
TOSTOP　向后台输出发送SIGTTOU信号

c_cc[NCCS]：控制字符，用于保存终端驱动程序中的特殊字符，如输入结束符等。c_cc中定义了如表5所示的控制字符。
　表5 c_cc支持的控制字符

宏　说 明　宏　说 明
VINTR　Interrupt字符　VEOL　附加的End-of-file字符
VQUIT　Quit字符　VTIME　非规范模式读取时的超时时间
VERASE　Erase字符　VSTOP　Stop字符
VKILL　Kill字符　VSTART　Start字符
VEOF　End-of-file字符　VSUSP　Suspend字符
VMIN　非规范模式读取时的最小字符数
tcsetattr函数用于设置终端的相关参数。参数fd为打开的终端文件描述符，参数optional_actions用于控制修改起作用的时间，而结构体termios_p中保存了要修改的参数。
　optional_actions可以取如下的值：
　TCSANOW：不等数据传输完毕就立即改变属性。
　TCSADRAIN：等待所有数据传输结束才改变属性。
　TCSAFLUSH：清空输入输出缓冲区才改变属性。
　错误信息：
　EBADF：非法的文件描述符。
　EINTR：tcsetattr函数调用被信号中断。
　EINVAL：参数optional_actions使用了非法值，或参数termios中使用了非法值。
　ENCTTY：非终端的文件描述符。

设置这个结构体很复杂，我这里就只说说常见的一些设置：

波特率设置

下面是修改波特率的代码：

```
struct  termios Opt;
tcgetattr(fd, &Opt);//tcgetattr函数用于获取与终端相关的参数。参数fd为终端的文件描述符，返回的结果保存在termios结构体中，该结构体一般包括如下的成员：
cfsetispeed(&Opt,B19200);     /*设置为19200Bps*/
cfsetospeed(&Opt,B19200);
tcsetattr(fd,TCANOW,&Opt);
/*
tcsetattr函数用于设置终端参数。函数在成功的时候返回0，失败的时候返回-1，并设置errno的值。参数fd为打开的终端文件描述符，参数
optional_actions用于控制修改起作用的时间，而结构体termios_p中保存了要修改的参数。optional_actions可以取如下的值。
 TCSANOW：不等数据传输完毕就立即改变属性。
 TCSADRAIN：等待所有数据传输结束才改变属性。
 TCSAFLUSH：清空输入输出缓冲区才改变属性。
*/
```

设置波特率的例子函数：

```
/**
*@brief  设置串口通信速率
*@param  fd     类型 int  打开串口的文件句柄
*@param  speed  类型 int  串口速度
*@return  void
*/
```

```
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++) { //计算数组中的数据个数
    if  (speed == name_arr[i]) {
      tcflush(fd, TCIOFLUSH);
/*

tcflush() 丢弃要写入引用的对象，但是尚未传输的数据，或者收到但是尚未读取的数据，取决于 queue_selector 的值：
  TCIFLUSH   刷新收到的数据但是不读
  TCOFLUSH   刷新写入的数据但是不传送
  TCIOFLUSH  同时刷新收到的数据但是不读，并且刷新写入的数据但是不传送

       通俗地说就是将输出缓冲器清空，把输入缓冲区清空。缓冲区里的数据都废弃。

*/
      cfsetispeed(&Opt, speed_arr[i]);
      cfsetospeed(&Opt, speed_arr[i]);
      status = tcsetattr(fd1, TCSANOW, &Opt);
      if  (status != 0) {
        perror("tcsetattr fd1");
        return;
      }
      tcflush(fd,TCIOFLUSH);
    }
  }
}
```

效验位和停止位的设置：

无效验	8位	Option.c_cflag &= ~PARENB; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS8;
奇效验(Odd)	7位	Option.c_cflag |= ~PARENB; Option.c_cflag &= ~PARODD; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS7;
偶效验(Even)	7位	Option.c_cflag &= ~PARENB; Option.c_cflag |= ~PARODD; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS7;
Space效验	7位	Option.c_cflag &= ~PARENB; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= &~CSIZE; Option.c_cflag |= CS8;

设置效验的函数：

```
/**
*@brief   设置串口数据位，停止位和效验位
*@param  fd     类型  int  打开的串口文件句柄
*@param  databits 类型  int 数据位   取值 为 7 或者8
*@param  stopbits 类型  int 停止位   取值为 1 或者2
*@param  parity  类型  int  效验类型 取值为N,E,O,,S
*/
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) /*设置数据位数*/
  {
 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); /* 设置为奇效验*/
  options.c_iflag |= INPCK;             /* Disnable parity checking */
     break;
 case 'e':
 case 'E':
  options.c_cflag |= PARENB;     /* Enable parity */
  options.c_cflag &= ~PARODD;    /* 转换为偶效验*/
  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);
   }
/* 设置停止位*/
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; /* 设置超时15 seconds*/
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);
}
```

需要注意的是:

如果不是开发终端之类的，只是串口传输数据，而不需要串口来处理，那么使用原始模式(Raw Mode)方式来通讯，设置方式如下：

```
options.c_lflag  &= ~(ICANON | ECHO | ECHOE | ISIG);  /*Input*/
options.c_oflag  &= ~OPOST;   /*Output*/
```

读写设备

设置好串口之后，读写串口就很容易了，把串口当作文件读写就是。

- 发送数据

```
char  buffer[1024];int    Length;int    nByte;nByte = write(fd, buffer ,Length)
```

- 读取串口数据

  使用文件操作read函数读取，如果设置为原始模式(Raw Mode)传输数据，那么read函数返回的字符数是实际串口收到的字符数。

  可以使用操作文件的函数来实现异步读取，如fcntl，或者select等来操作。

```
char  buff[1024];int    Len;int  readByte = read(fd,buff,Len);
```

关闭串口

关闭串口就是关闭文件。

```
close(fd);
```

例子

下面是一个简单的读取串口数据的例子，使用了上面定义的一些函数和头文件

```
/*****************************************************************
代码说明:使用串口二测试的，发送的数据是字符，
但是没有发送字符串结束符号，所以接收到后，后面加上了结束符号。
我测试使用的是单片机发送数据到第二个串口，测试通过。
*****************************************************************/
#define FALSE  -1
#define TRUE   0
/*****************************************************************/
int OpenDev(char *Dev)       //dev 可以是USB 也可以是串口
{
 int fd = open( Dev, O_RDWR );         //| O_NOCTTY | O_NDELAY
 if (-1 == fd)
    {
  perror("Can't Open Serial Port");
    return  -1;
    }
  else
    return  fd;
}
int main(int argc, char **argv){
  int  fd;
  int  nread;
  char buff[512];
  char *dev  = "/dev/ttyS1"; //串口二
  fd = OpenDev(dev);
  set_speed(fd,19200);
  if (set_Parity(fd,8,1,'N') == FALSE)  {
   printf("Set Parity Error\n");
    exit  (0);
       }
while (1) //循环读取数据
{
 while((nread = read(fd, buff, 512))>0)
    {
   printf("\nLen %d\n",nread);
   buff[nread+1] = '\0';
   printf( "\n%s", buff);
       }
}
 //close(fd);
 // exit (0);
}
```

总代码

```
[cpp]
```

```c
#include     <stdio.h>      /*标准输入输出定义*/
#include     <stdlib.h>     /*标准函数库定义*/
#include     <unistd.h>     /*Unix标准函数定义*/
#include     <sys/types.h>  /**/
#include     <sys/stat.h>   /**/
#include     <fcntl.h>      /*文件控制定义*/
#include     <termios.h>    /*PPSIX终端控制定义*/
#include     <errno.h>      /*错误号定义*/

/***@brief  设置串口通信速率
*@param  fd     类型 int  打开串口的文件句柄
*@param  speed  类型 int  串口速度
*@return  void*/

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)   //Linux 下串口USB等设备通信编程入门
2中有终端属性的获取设置函数
{
  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);//Update the options and do it NOW
        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);
   }
}
/**
*@brief   设置串口数据位, 停止位和效验位
*@param  fd     类型  int  打开的串口文件句柄*
*@param  databits 类型  int 数据位   取值 为 7 或者8    数据位为7位或8位
*@param  stopbits 类型  int 停止位   取值为 1 或者2*    停止位为1或2位
*@param  parity  类型  int  效验类型 取值为N,E,O,,S     N->无奇偶校验,
O->奇校验 E->为偶校验,
*/
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) /*设置数据位数*/
  {
    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);  /* 设置为奇效验*/
        options.c_iflag |= INPCK;             /* Disnable parity che
cking */
        break;
    case 'e':
    case 'E':
        options.c_cflag |= PARENB;     /* Enable parity */
        options.c_cflag &= ~PARODD;   /* 转换为偶效验*/
        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);
        }
  /* 设置停止位*/
  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;
    options.c_cc[VTIME] = 150; // 15 seconds
    options.c_cc[VMIN] = 0;

  tcflush(fd,TCIFLUSH); /* Update the options and do it NOW */
  if (tcsetattr(fd,TCSANOW,&options) != 0)
    {
        perror("SetupSerial 3");
        return (FALSE);
    }
  return (TRUE);
 }
/**
*@breif 打开串口
*/
int OpenDev(char *Dev)
{
int fd = open( Dev, O_RDWR );         //| O_NOCTTY | O_NDELAY
    if (-1 == fd)
        { /*设置数据位数*/
            perror("Can't Open Serial Port");
            return -1;
        }
    else
    return fd;

}
/**
*@breif     main()
*/
int main(int argc, char **argv)
{
    int fd;
    int nread;
    char buff[512];
```

```
    char *dev ="/dev/ttyS1";
    fd = OpenDev(dev);
    if (fd>0)
    set_speed(fd,19200);
    else
        {
        printf("Can't Open Serial Port!\n");
        exit(0);
        }
  if (set_Parity(fd,8,1,'N')== FALSE)  // 8位数据，非两位的停止位，不使用奇偶校验 ，不允许输入奇偶校验

  {
    printf("Set Parity Error\n");
    exit(1);
  }
  while(1)
    {
        while((nread = read(fd,buff,512))>0)
        {
            printf("\nLen %d\n",nread);
            buff[nread+1]='\0';
            printf("\n%s",buff);
        }
    }
    //close(fd);
    //exit(0);
}
```

Oracle创建表时Storage参数具体含义 03-03
本文通过图表和实例的阐述在Oracle数据库创建新表时Storage的参数具体含义。

C代码设置串口的各个参数总结 05-28
打开串口、设置波特率、数据位、校验位、停止位、流控设置、控制字符、关闭等涉及到的各个参数含义

抢沙发
优质评论可以帮助作者获得更高权重
评论

智能家居中的物联网网关的可信计算平台模块(TPM)设计... 8-4
{B115200, B38400, B19200, B9600, B4800, B2400, B1200, B300, B38400, B19200, B9600, B4800, B2400, B1200, B300, };intname_arr[] = {115200,3...

linux 串口通信详解_一技旁身_linux的串口通信 7-19
options.c_cflag |= CS8; // 数据位为 8 ,CS7 for 7 options.c_cflag &= ~CSTOPB; // 一位停止位, 两位停止为 |= CSTOPB options.c_cflag &= ~PARENB; // ...

串口通信每个参数代表什么 复杂的程序员 2823
串口的BaudRate波特率表示数据传输的速率。 串口的PortName表示用那个口发送接收数据。 串口的DataBits表示能表示的数据最大支持多少位。

Linux 下串口USB等设备通信编程入门1 遁法自然 1万+
此文章主要以串口为例子进行讲解，USB只需要修改打开的终端设备就可以 串口简介 串行口是计算机一种常用的接口，具有连接线少，通讯简单，得到广...

vmstat/top/w查看ubuntu内存CPU使用情况以及参数解释... 8-27
procs部分的解释 r 列表示运行和等待cpu时间片的进程数,如果长期大于1,说明cpu不足,需要增加cpu。 b 列表示在等待资源的进程数,比如正在等待I/O、或...

计算机类顶级会议排名+投稿经验_Beryl的博客 8-24
ADBIS Symposium on Advances in DB and Information Systems B ADC Australasian Database Conference B ADCS Australasian Document Computing...

linux 下串口通信 mj5742356的专栏 1万+
以前跟着做过VxWorks的开发，主要通信方式是串口，因为底层BSP包已经做好了，串口通信非常简单。后来接触Linux，在一块OK6410上跑Linux串口通...

树莓派串口通信配置、测试(全程) qq_43765237的博客 1535
一、树莓派串口配置修改 1.将树莓派的硬件串口与mini串口默认映射对换 简单来说：硬件串口由硬件实现，有单独波特率时钟源，性能高，可靠，而mini...

HTTP 请求头各参数具体含义 qq_42928070的博客 706
https://blog.csdn.net/xiaochengyihe/article/details/80910913

sort -k参数具体含义 dengjh_213的博客 8764
[FStart [.CStart]] [Modifier] [. [FEnd [.CEnd]][Modifier]] FStart，从第几列开始 .CStart，从该列第几个字符开始 Modifier，只n、r等其他参数 FEn...

串口设置（波特率、数据位、校验位、停止位） 贰兜屋 5212
串口终端设备的接口属性如下： struct termios { tcflag_t c_cflag; //控制标志 tcflag_t c_iflag; //输入标志 tcflag_t c_oflag; //输出标志 tcflag_t c_lflag; //本地...

不同类型指针变量的具体含义 最新发布 08-03
在嵌入式系统中，对单片机内寄存器的操作经常使用其地址，这样使程序更加简洁、直观，但地址具体是什么类型的数据？它和它指向的数据类型有什么...

Linux串口通讯（mark, space校验方式的实现） DiegoTJ的专栏 4206
前一阵子因为工作需要摸索的一些linux下得串口通信，总结下结果， 有点乱。。。主要针对linux串口校验方式mark, space的摸索。。。 参考文档： 文档...

【Tools】Linux串口设备调试技巧（sftty） 只要思想不滑坡，想法总比问题多。 1704
串口设备是linux系统中最基本的设备之一，在嵌入式linux开发中几乎是必不可少的。由于串口使用简单、广泛，除了使用一路串口作为调试终端输出外，...

stty 命令说明及使用讲解 Alan0521的专栏 1万+
stty 命令说明及使用讲解 UNIX系统的命令很多，但是巧妙使用命令的方法更多。随着经验的积累和观察学习其他用户的实践，我们也可学会解决特殊问...

智能家居中的物联网网关的可信计算平台模块（TPM）设计 架构设计 216
摘要： 随着智能家居的普及，安全性问题的研究已成当务之急。针对物联网网关自身的易受攻击性和网络传输过程中的信息，我们分别采用SHA-1和AES...

Http Header一览 baidu1966的博客 112
Requests Header | Http Header Header 解释 示例 Accept 指定客户端能够接收的内容类型 Accept: text/plain, text/html Accept-Charset 浏览器可以接受的...

linux中top命令详解 weixin_33850015的博客 237
在linux的top命令里的cpu信息是什么意思呢？ Cpu(s): 62.1% us, 15.9% sy,0.1% ni, 19.4% id,2.0% wa,0.1% hi,0.4% si Mem: 8247956k total,8232004k u...

英文投稿的一点经验【转载】 热门推荐 jianwushuang的专栏 1万+

英文投稿的一点经验【转载】
1. 首先一定要注意杂志的发表范围, 超出范围的千万别投,要不就是浪费时间;另外，每个杂志都有他们的具体格式...

进程查看及命令使用-htop/dstat/top/ps命令 weixin_34406796的博客 52
进程是linux用来表示正在运行的程序的一种抽象概念，程序内存的使用，处理器时间和I/O资源就是通过这个对象进行管理和监视的。一个程序要先运行在...

believe209
码龄15年 暂无认证

81	6万+	55万+	180万+	
原创	周排名	总排名	访问	等级
1万+	300	301	99	1101
积分	粉丝	获赞	评论	收藏

私信 关注

搜博主文章

热门文章
windows下添加路由 116741
IP数据报格式详解 72006
C#播放声音【六种方法】 49456
tftp命令使用说明 49254
Linux 下使用USB 网络 48057

最新评论
linux下bluetooth编程（三）HCI层编程
chaoshuaihaohao: 这些图片原文档有链接吗？
linux下基于QT的串口程序
zhangshuyang321: 串口参数设置部分有一个错了，是myCom->setDataBits(QSeria...
IP数据报格式详解
wupeng0: 博主不是说了瓶颈是从A到B路由所经过的网络的最小MTU
用C写蓝牙通讯程序：扫描、读取、发送
MFSVxiaomo: 不可以的
用C写蓝牙通讯程序：扫描、读取、发送
Simba的风车乐: 哥，win下可以操吗

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
Linux环境下的GDB调试方法
【C#基础知识】获取网卡的ip地址和MAC
ARP协议工作原理[同网段及跨网段]

2020年 1篇 2017年 33篇
2016年 99篇 2015年 130篇
2014年 26篇 2013年 5篇
2012年 7篇

believe209 关注 2 0 12 专栏目录

疯狂盲盒 举报