

串口部分实验报告

一、实验 4.2——Linux 操作系统平台

实验目的：

- 1、了解 Linux 系统文件系统的基本组织；
- 2、了解 Linux 基本的多用户权限系统；
- 3、熟练使用 ls、cd、cat、more、sudo、gcc、vim 等基本命令；
- 4、会使用 ls 和 chmod 命令查看和修改文件权限

实验过程及分析：

第一步：下载好实验平台给出的 deb 文件，进行安装操作

```
(正在读取数据库 ... 系统当前共安装有 189378 个文件和目录。)  
准备解压 firstrun.deb ...  
正在解压 firstrun-package (1.0-1) 并覆盖 (1.0-1) ...  
正在设置 firstrun-package (1.0-1) ...
```

第二步：安装成功后，下一步操作是运行根目录下的/gettips 可执行程序。在当前窗口继续输入/gettips 后回车后，显示屏上输出了一个文件路径

```
ljq@ljq-virtual-machine:~/桌面$ /gettips  
/usr/bin/tianma
```

第三步：在第二步给出的目录中存在一个文件，由于此文件可能被隐藏，于是考虑使用 ls -a 命令，此命令可以显示出当前文件夹下所有文件，不过这里需要注意的是此操作需要切换到超级用户。于是先输入 sudo -s，切换到超级用户后，再进行 ls -a 操作，如下图所示

```
ljq@ljq-virtual-machine:~/桌面$ sudo -s  
root@ljq-virtual-machine:/home/ljq/桌面# cd /usr/bin/tianma  
root@ljq-virtual-machine:/usr/bin/tianma# ls -a  
.. .puzzle.txt
```

要显示该文件内容，可以使用 cat 操作，也可以直接使用 gedit 打开。

打开后能看见文件内容为：3350309043

第四步：使用实验平台给出的命令将文件内容提交到实验服务器，其中 v 为文件内容，id 为学号，于是在终端继续输入：

```
curl"132.232.98.70:6363/check?id=202109070105&v=3350309043"
```

由于 curl 之前没有安装，所以需要先用 apt install curl 安装好 curl 后此操作才能执行成功。

安装好 curl 后，重新输入该命令，回车，显示屏显示 OK，即操作成功执行

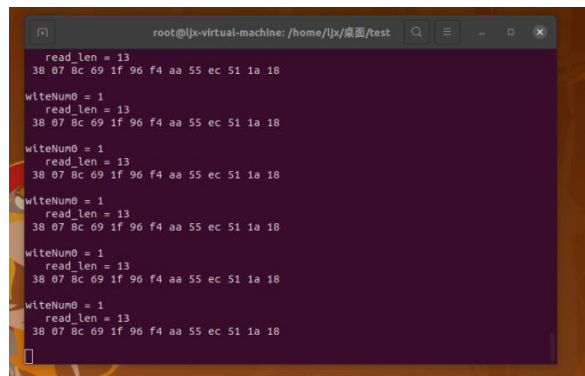
第五步：编写 C 语言程序，读取第 4 步所找到文件中的内容并转换成十六进制数输出到屏幕上，编写的 C 语言代码如下：

```
1 #include<stdio.h>  
2 #include<stdlib.h>  
3 char buffer[33];  
4 char *in_hex(int num)  
5 {  
6     sprintf(buffer, "%x", num);  
7     return (buffer);  
8 }  
9 int main()  
10 {  
11     int n;  
12     char *hex_str;  
13     FILE *file=fopen("/usr/bin/tianma/.puzzle.txt","r");  
14     if(file==NULL)  
15     {  
16         printf("open error!\n");  
17         return 0;  
18     }  
19     char buf[1024]={0};  
20     fgets(buf,1024,file);  
21     n=atoi(buf);  
22     hex_str=in_hex(n);  
23     printf("Hexadecimal number:%s\n",hex_str);  
24     fclose(file);  
25     return 0;  
26 }  
27
```

由于需要使用 C 语言打开文件进行读写，而文件夹可能存在权限限制，所以我们需要先将文件夹以及文件夹内的文件修改为可读可写可执行，使用 chmod 777 .puzzle.txt 与

片机最为接近的波特率为 115200。

第三步：在 com.c 中将波特率修改为 115200，接着在 Linux 下对 main.c 编译运行，生成可执行文件后，使用 ./命令执行，输出结果如下：



```
root@ljk-virtual-machine: /home/ljk/桌面/test
read_len = 13
38 07 8c 69 1f 96 f4 aa 55 ec 51 1a 18

writeNum0 = 1
read_len = 13
38 07 8c 69 1f 96 f4 aa 55 ec 51 1a 18

writeNum0 = 1
read_len = 13
38 07 8c 69 1f 96 f4 aa 55 ec 51 1a 18

writeNum0 = 1
read_len = 13
38 07 8c 69 1f 96 f4 aa 55 ec 51 1a 18

writeNum0 = 1
read_len = 13
38 07 8c 69 1f 96 f4 aa 55 ec 51 1a 18

writeNum0 = 1
read_len = 13
38 07 8c 69 1f 96 f4 aa 55 ec 51 1a 18
```

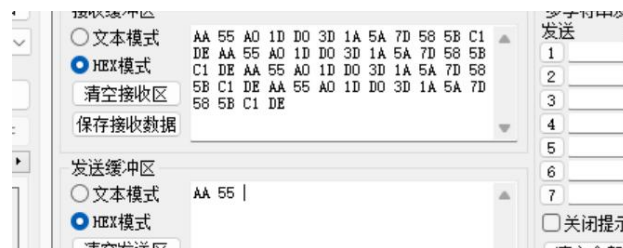
因此，序列号为 ec511a1838078c691f96f4

第四步：使用 curl 命令将结果提交，回车后显示 OK

三、实验 4.4——计算机串口数据收发与测量

编写代码：

1、使用 STC-ISP 程序串口助手模拟数据收发与测量，打开串口助手得到板子的序列号为：a01dd03d1a5a7d585bc1de



2、编写代码实现串口数据的收发与测量，注意将 com.c 中波特率修改为 1200.

```

1 #include<stdio.h>
2 #include "com.h"
3 #include "com.c"
4 #include <sys/epoll.h>
5
6 #define LONG_STRING_LEN 12 //每次读的字符串的长度 取决于AA 55 后面的数字 每个人不一样
7 #define CODE_LEN 6 //密码长度 (AA 55 + 四位密码)
8 #define ID_LEN 15 //学号长度15
9 #define MAX_REPEAT_TIMES 3 //重复N次则判定为读到最后 防止误判
10
11
12 //用来打印字符串 参数为字符串和长度
13 void string_print(unsigned char *s, int len)
14 {
15     int i;
16     for (i = 0; i < len; i++)
17     {
18         printf("%02X ", s[i]);
19     }
20     printf("\n");
21 }
22
23 //重写read函数 读取固定长度的字符串 (comRead和read只会返回实际读取的数目)
24 int read_n_bytes(int fd, char* buf, int n)
25 {
26     /*read_len用来记录每次读到的长度, len用来记录当前已经读取的长度*/
27     int read_len = 0, len = 0;
28
29     /*read_buf用来存储读到的数据*/
30     char* read_buf = buf;
31     while(len < n)
32     {
33         if((read_len = read(fd, read_buf, n)) > 0)
34         {
35             len += read_len;
36             read_buf += read_len;
37             read_len = 0;
38         }
39
40         //休眠时间视情况而定, 可以参考波特率来确定数量级
41         usleep(20);
42     }
43     return len;
44 }
45
46
47 int main(void)
48 {
49     unsigned char ID[15] = {0xAA, 0x55, 0x02, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x03, 0x01, 0x09, 0}; //学号
50     unsigned char long_string[64] = {0}; //截取含有密码的字符串
51     unsigned char last_code[7] = {0}; //保留最近的一次密码 用来对比是否已经读到了最后一个密码
52     unsigned char code[7] = {0}; //每次的密码 开头为AA 55 ;接着四位为密码
53     code[0] = 0xAA;
54     code[1] = 0x55;
55     last_code[0] = 0xAA;
56     last_code[1] = 0x55;
57     int read_len = 0; //读到字符的数量
58     int code_num = 0; //记录读到的密码的数量
59     int flag = 0; //标志位: 读到最后一个密码
60     int index=0; //遍历用
61     int repeat_times=0; //密码重复次数
62
63     fd = openSerial("/dev/ttyUSB5"); //打开串口, ttyUSB0是串口文件
64     if (fd < 0)
65     {
66         printf("open com fail!\n");
67         return 0;
68     }
69     EpollInit(fd); //初始化终端事件触发函数epoll, 设置要监听的事件及相关参数等
70
71     write(fd, ID, ID_LEN); //第一次先写入学号
72     //串口, 学号字符串, 长度
73     usleep(200000); //等待写入完成 200ms
74
75     printf("write ID successful\n\n"); //首次写入学号成功
76
77     while (long_string[0]!=0xAA || long_string[1]!=0x55) //读到第一个AA 55 用来对齐
78     {
79         read_len = read_n_bytes(fd, long_string, LONG_STRING_LEN); //读取
80         tcflush(fd, TCIFLUSH); //清空输入缓存
81     }
82
83     string_print(long_string, LONG_STRING_LEN);
84
85

```

```

86  int k=1;
87  while(1){
88      read_len = read_n_bytes(fd,long_string, LONG_STRING_LEN); //读取以AA 55开始的字符串
89      code[2] = long_string[8];
90      code[3] = long_string[9];
91      code[4] = long_string[10];
92      code[5] = long_string[11];
93
94      write(fd,code, CODE_LEN); //密码写回
95
96      //判断时候为最后一个密码
97      flag=0;
98      for (index = 2; index <= 5; index++) //对比前后两次密码
99      {
100         if (code[index] != last_code[index])
101         {
102             flag = 1;
103             break;
104         }
105     }
106
107     if (flag) //前后密码不相等 有效密码
108     {
109         repeat_times=0;
110         code_num++;
111     }else { //读到最后一个密码
112         break;
113     }
114
115     // else :记录新的密码
116     for (index = 2; index <= 5; index++)
117         last_code[index] = code[index];
118
119     printf("得到的当前密码: ");
120     string_print(code, CODE_LEN);
121     printf("已经读到的密码数: %d\n\n", code_num);
122 }
123
124
125 printf("读取到第 %d 个密码,最后的密码是: ",code_num);
126 string_print(&last_code[2], CODE_LEN-2); //输出最后4位密码
127 printf("\n\n");
128
129 close(epid);
130 close(fd);
131
132 return 0;
133 }

```

3、使用 gcc 将代码编译链接生成可执行文件，运行使用代码与单片机进行数据交互，得到的最后一个密码是：

```

小李得到的当前密码： AA 55 51 5A C2 DE
已经读到的密码数： 256

读取到第 256 个密码,最后的密码是： 51 5A C2 DE

```

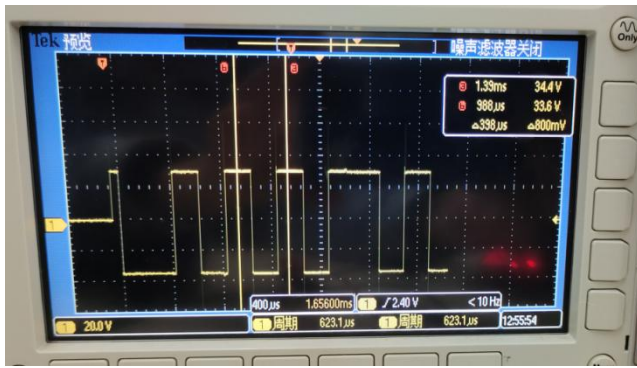
4、使用 curl 命令将学号、板子序列号以及最后一个密码提交到平台上：

四、实验 4.5——RS485 信号的测量

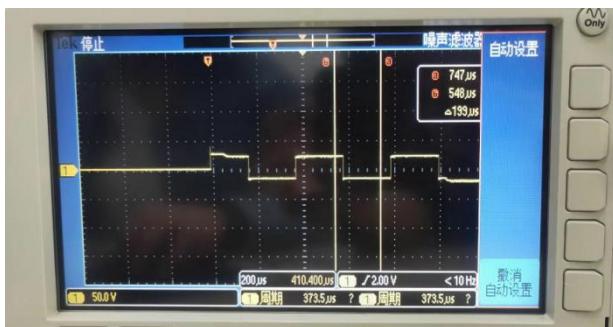
- 【实验目的】 1、熟练使用 linux 下 io 函数 read、write 和 epoll 等
2、熟悉 RS485 串口的信号特点

【实验过程】

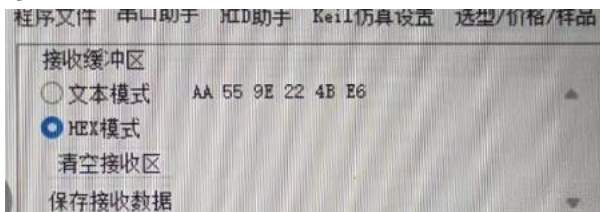
第一步、本实验需要两位同学合作完成，我的板子首先作为 B 板，同学的板作为 A 板，首先使用示波器观察 A 板的波特率，示波器观察得到的图像如下所示，求得相近的波特率为 4800。



第二步、为使得 A 板 B 板能接收到正确的数据，我们需要确保两个板子的波特率相同。下面开始调整 B 板的波特率，拨动 B 板上的摇杆同时观察示波器，使得其波特率与 A 板相同即可。



第三步、向 B 板下载 B.hex 文件，与同学下载好程序的 A 板通过杜邦线连接，同学按下 K 键后 A 板向 B 板发送序列号信息，通过串口助手能看到 A 板的序列号，记下该序列号：9e224be6



第四步、通过 C 语言编程，调用 write 函数向串口发送读取密码命令，编写的代码如下：

```
1 #include <stdio.h>
2 #include "com.h"
3 #include "com.c"
4 int main(void)
5 {
6     unsigned char tmp[18] = {0xAA,0x55,0x9E,0x22,0x4B,0xE6,0x02,0x06,0x02,0x00,0x00,0x08,0x01,0x06,0x03,0x01,0x09};
7     unsigned char buffer[6];
8     fd = openSerial("/dev/ttyUSB2"); //打开串口，ttyUSB1是串口文件
9
10    write(fd,tmp,19);
11    read(fd,buffer,6);
12    int i;
13    for(i=0;i<6;i++)
14        printf("%02x",buffer[i]);
15    return 0;
16 }
```

第五步、运行该程序，得到输出结果为：28ee9c4b，使用 curl 命令将序列号以及密码提交到平台上，回车显示 OK

与同学交换，后又作为 A 板，测得波特率与序列号，此处不再赘述。

【实验总结】通过本次实验，我了解了 RS485 串口的信号特点以及工作原理，获悉了串行通信与同步通信的区别，本次实验是本章中第一次由两位同学合作完成的实验，本次实验也帮助我们培养合作意识、提高合作交流能力。

五、实验 4.6——RS485 总线数据收发

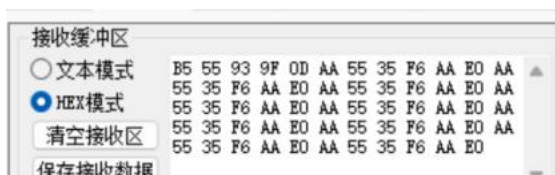
【实验目的】 1、熟练使用 Linux 下 io 函数 read、write 和 epoll 等

2、熟练处理流式通信数据

3、理解 485 总线的冲突问题

【实验过程】

第一步、选定自己的板子首先作为 B 板，下载好 B 板对应的程序，同学的板子作为 A 板，下载好程序后与自己的板子通过杜邦线相连，由于 A 板发送序列号的波特率固定为 1200，不必再使用示波器测量，在本机上可以使用串口助手指定波特率为 1200，点击打开串口，即可接收到 A 板发送的序列号，如下所示：A 板的序列号为 35f6aaed。



第二步、通过 B 板串口向 RS485 总线上写入自己的学号，A 板接收到学号后能返回第一串密码，B 板需要在 150ms 内将密码发送回去，否则会造成总线冲突，因为 RS485 只有一个逻辑信号，在同一个时刻只能有一个主体进行数据发送。按照该要求，需要进行编程实现数据的收发。由于密码中间会有缓冲字节，即样式为 AA 55 +（未知长度的缓冲字节）+ 四字节密码，我们可以先使用串口助手观察密码缓冲字节长度，通过观察，缓冲部分为两个字节。

第三步、按照要求编程，在实验 3.4 代码的基础上做一些改动即可，修改读的字符串长度为 8 个字节，修改 read 函数，将休眠 sleep 删掉，其余部分基本不变，代码如下：

```

1 #include<stdio.h>
2 #include "com.h"
3 #include "com.c"
4 #include <sys/epoll.h>
5
6 #define LONG_STRING_LEN 8 //每次读的字符串的长度 取决于AA 55 后面的数字 每个人不一样
7 #define CODE_LEN 6 //密码长度 (AA 55 + 四位密码)
8 #define ID_LEN 15 //学号长度15
9 #define MAX_REPEAT_TIMES 3 //重复N次则判定为读到最后 防止误判
10
11
12 //用来打印字符串 参数为字符串和长度
13 void string_print(unsigned char *s, int len)
14 {
15     int i;
16     for (i = 0; i < len; i++)
17     {
18         printf("%02X ", s[i]);
19     }
20     printf("\n");
21 }
22
23 //重写read函数 读取固定长度的字符串 (ComRead和read只会返回实际读取的数目)
24 int read_n_bytes(int fd, char* buf, int n)
25 {
26     /*read_len用来记录每次读到的长度, len用来记录当前已经读取的长度*/
27     int read_len = 0, len = 0;
28
29     /*read_buf用来存储读到的数据*/
30     char* read_buf = buf;
31     while(len < n)
32     {
33         if((read_len = read(fd, read_buf, n)) > 0)
34         {
35             memcpy(read_buf, read_buf, read_len);
36             read_buf += read_len;
37             len += read_len;
38         }
39     }
40 }

```

```

32 {
33     if((read_len = read(fd,read_buf,n)) > 0)
34     {
35         len += read_len;
36         read_buf += read_len;
37         read_len = 0;
38     }
39
40     //休眠时间视情况而定，可以参考波特率来确定数量级
41     //usleep(20);
42 }
43 return n;
44 }
45
46 int main(void)
47 {
48     unsigned char ID[15] = {0xAA, 0x55, 0x02, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x03, 0x01, 0x09, 0}; //学号
49     unsigned char long_string[64] = {0}; //截取含有密码的字符串
50     unsigned char last_code[7] = {0}; //保留最近的一次密码 用来对比是否已经读到了最后一个密码
51     unsigned char code[7] = {0}; //每次的密码 开头为AA 55 ;接着四位为密码
52     code[0] = 0xAA;
53     code[1] = 0x55;
54     last_code[0] = 0xAA;
55     last_code[1] = 0x55;
56     int read_len = 0; //读到字符的数量
57     int code_num = 0; //记录读到的密码的数量
58     int flag = 0; //标志位：读到最后一个密码
59     int index=0; //遍历用
60     int repeat_times=0; //密码重复次数
61
62     fd = openSerial("/dev/ttyUSB7"); //打开串口，ttyUSB7是串口文件
63     if (fd < 0)
64     {
65         printf("open com fail!\n");
66         return 0;
67     }
68
69     EpollInit(fd); //初始化终端事件触发函数epoll,设置要监听的事件及相关参数等
70
71     write(fd, ID, ID_LEN); //第一次先写入学号
72     //串口，学号字符串，长度
73
74     usleep(200000); //等待写入完成 200us
75
76     printf("write ID successful\n"); //首次写入学号成功
77
78     while(1){
79         while (long_string[0]!=0xAA) //读到第一个AA 55 用来对齐
80         {
81             read_len = read_n_bytes(fd,long_string,1); //读取
82             //tcflush(fd, TCIFLUSH); //清空输入缓存
83         }
84
85         read_len = read_n_bytes(fd,long_string, LONG_STRING_LEN-1); //读取以AA 55开始的字符串
86
87         code[2] = long_string[3];
88         code[3] = long_string[4];
89         code[4] = long_string[5];
90         code[5] = long_string[6];
91
92         write(fd,code, CODE_LEN); //密码写回
93
94         //printf("得到的当前密码: ");
95         string_print(code, CODE_LEN);
96         //printf("已经读到的密码数: %d\n", code_num);
97     }
98
99     printf("读取到第 %d 个密码,最后的密码是: ",code_num);
100     string_print(&last_code[2], CODE_LEN-2); //输出最后4位密码
101     printf("\n\n");
102
103     close(epid);
104     close(fd);
105
106     return 0;
107 }

```

第五步、运行该程序，得到该程序读取到的最后一串密码，如下图为 942e57a3

AA 55 94 2E 57 A3

第六步、使用 curl 命令将序列号、密码以及学号提交到平台上，回车显示当前读取的密码数为 240。

【实验总结】本次实验就像 4.6 与 4.4 的结合版，综合考察了 RS485 的信号特点与 C 语

言编程实现收发数据。通过实验进一步熟悉了串口通信的方法, 以及使用 `write`, `read` 等函数进行串口数据进行读写。了解了 RS485 接口的使用, 通过 RS485 实现了两板的通信和数据传输。认识了 RS485 的半双工通信模式, 在程序运行时, 有些产生的密码是错误的, 即发生了冲突得到了错误数据。对于程序的编写, 能够通过 c 语言程序实现功能, 但是效果不好, 还需要进一步学习。后续还需要加深理解, 提高代码的性能以及编写相关代码的能力。