**GEC6818开发板（教学板）**

arm公司：主要业务芯片的设计

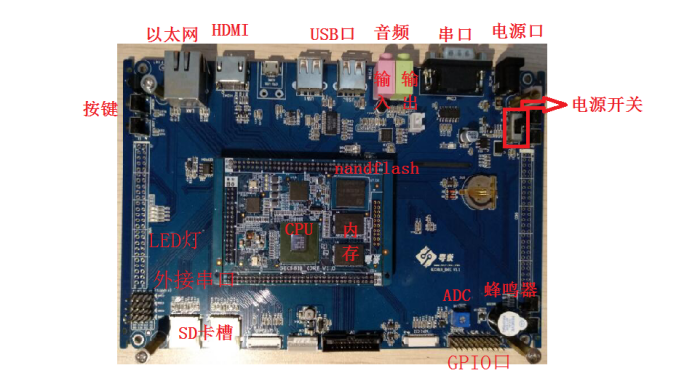
A系列： （高端）这类芯片是运行在很高的时钟频率，可以支持完整的操作系统（Linux、Windows）

M系列：（中端）主要应用于硬盘控制器，网络设备、打印机、汽车用：安全气囊、辅助驾驶、一键泊车、仪表盘，M系列是A系列的阉割，只能完成部分的功能

R系列：最低端的处理器，主要应用于工业生产，基础消费型电子设备（MP3、MP4），主要是偏向于控制方面（处理性能很弱），简单理解：高级的单片机。

三星公司根据arm设计的芯片：cortex-A53设计了一款CPU：S5P6818

粤嵌根据这款CPU设计了一款教学板：GEC6818开发板



核心板：

CPU：S5P6818

运行内存（RAM）：512MB\*2 = 1GB

存储(ROM)： 8GB

**6818的使用**

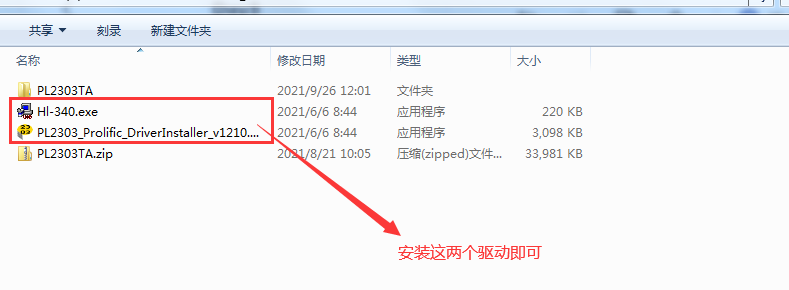
**6818的链接**

接线：电源线和串口线

安装串口驱动：

PL2303\_Prolific\_DriverInstaller\_v1210.exe

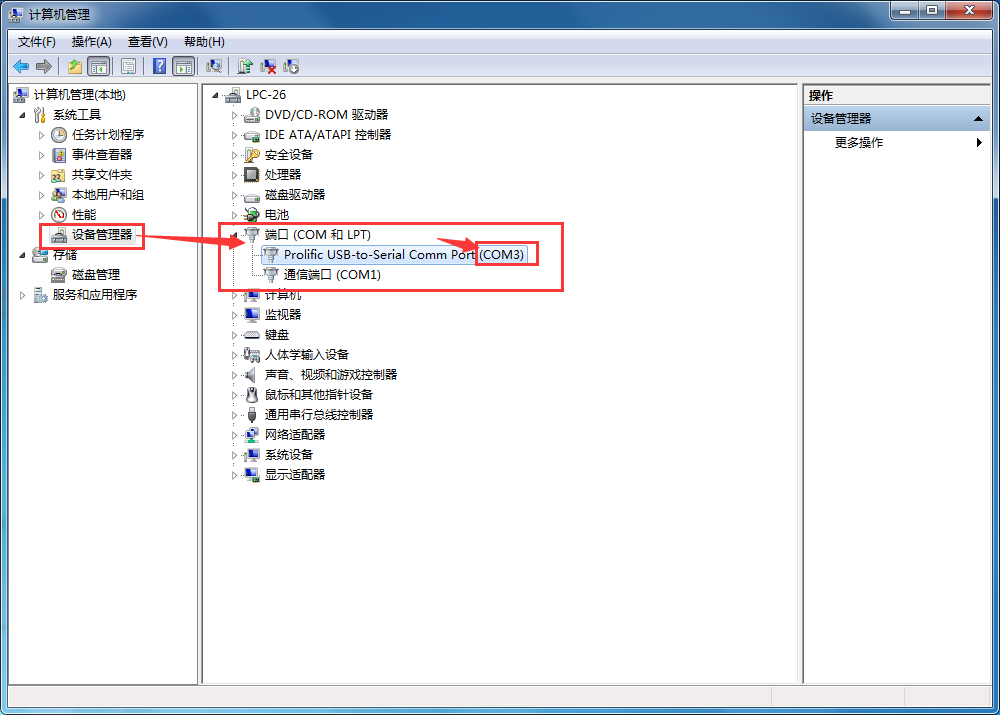
Hl-340.exe(可以不装）



检查串口是否被识别：

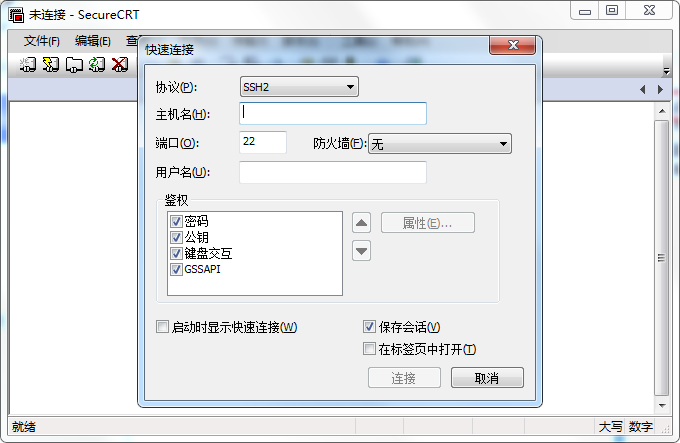
右键开始菜单--》设备管理器

你们不一定是COM3，而是随机的，以你们电脑自己读取到的为主



通信

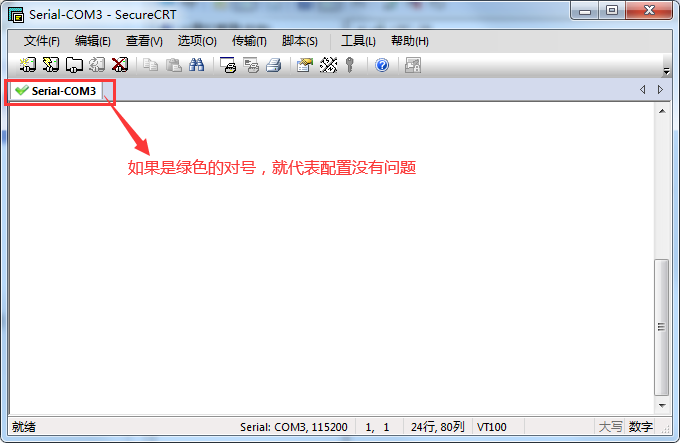
打开SecureCRT.exe程序



配置信息



检查配置是否成功



开机： 如果能正常弹数据，并且没有乱码，就代表没问题

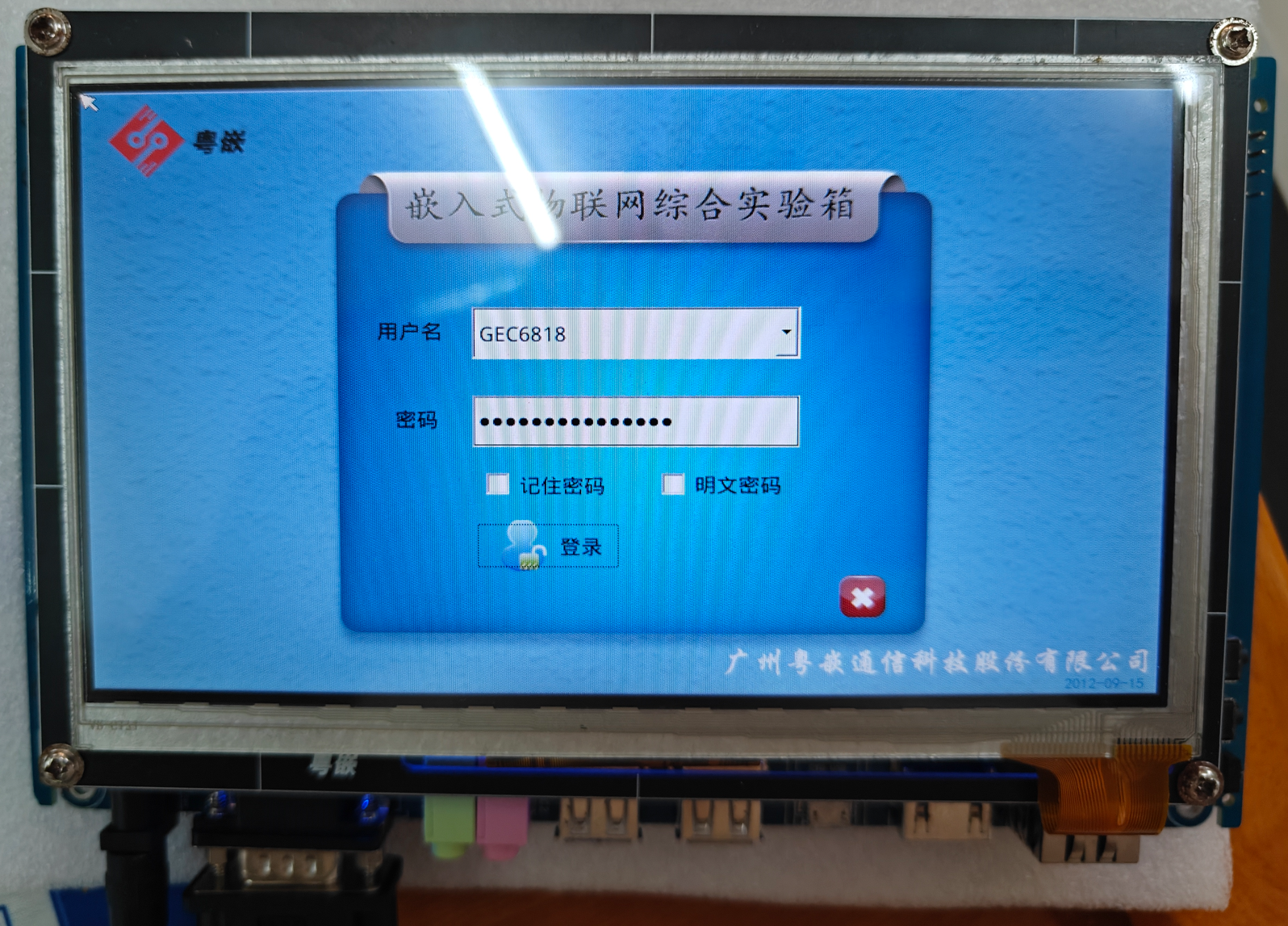
注意：不用管屏幕上显示了什么，只要屏幕可以正常亮就行了

如果屏幕上显示的是物联网试验箱的。

两种方法取消：

1. Ctrl+C取消（建立在能正常链接的基础上）

2. 点击屏幕上面右下角那个红色的X



**6818开发板传输文件**

终极目标：

在Windows下写代码

在Linux下编译代码生成程序

在开发板上运行程序

使用rx命令进行传输文件

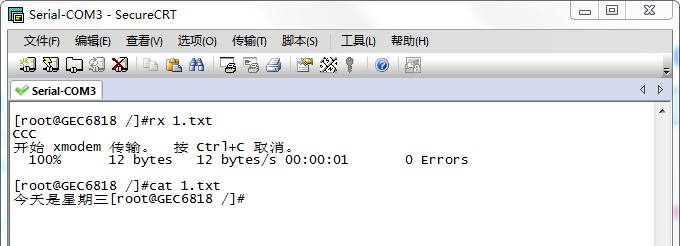
传输文件：其实就是把电脑中的某个文件复制到开发板中

　rx + 需要传输的文件的名字（把文件的后缀名加上） + 回车

例如： rx 1.txt

方法2：菜单栏 --》 传输 --》 发送xmodem --》 文件所在的位置 --》确定

要求：在开发板上的操作



开发板上运行程序

电脑上：gcc main.c -o main

gcc: 编译工具包， 编译生成的程序是x86-64架构

main.c:需要被编译的对象

-o:output输出二进制文件（程序）

main：二进制文件的名字

流程：

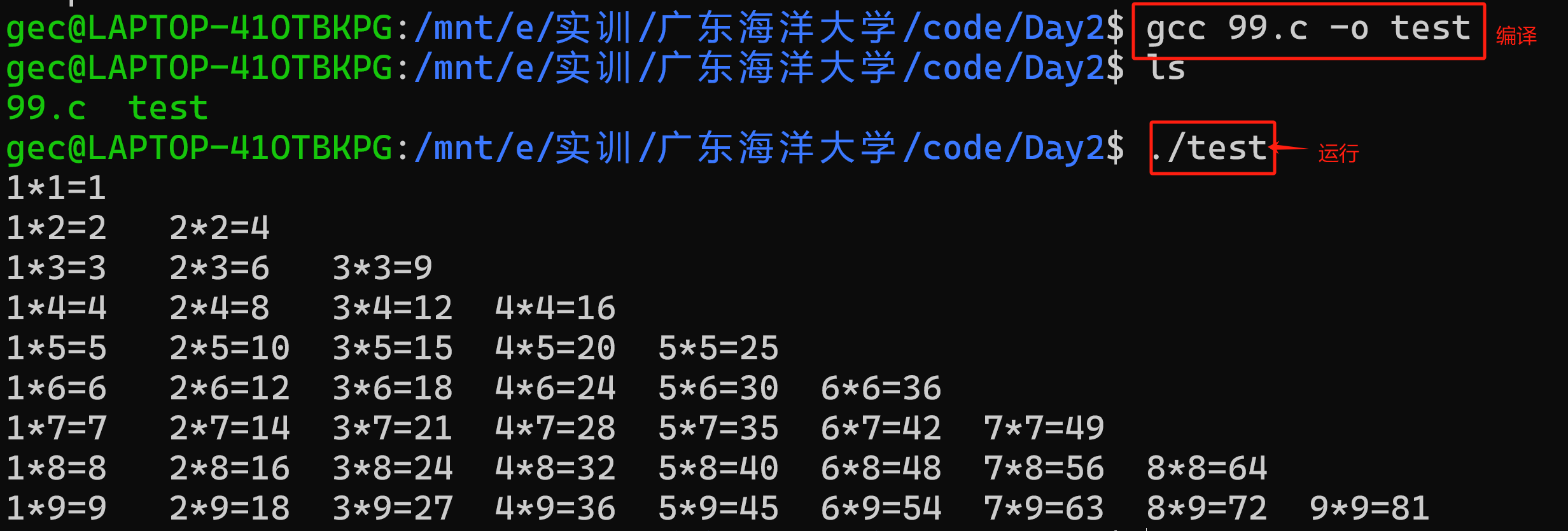
创建.c文件

写代码

保存

gcc编译

./运行



为什么使用arm-linux-gcc：

我们现在手上的设备是ARM的芯片，由于芯片的指令集有所不同，所以arm芯片只能运行arm架构的程序，而我们使用的gcc编译出来的是x86-64架构的，所以不能直接拿来用。

解决方法： 换一个编译工具包来编译代码：arm-linux-gcc

编译命令：arm-linux-gcc main.c -o main

总结：

如果要在电脑上运行程序，我们要是有gcc编译代码；

如果要在开发板上运行程序，我们要是有arm-linux-gcc编译代码

拓展：

芯片的指令集是一组可以由处理器执行的指令。这些指令可以包括算术运算、逻辑运算、数据传输、控制流程等操作。不同的芯片可能具有不同的指令集，例如x86架构的处理器和ARM架构的处理器具有不同的指令集。指令集的设计可以影响处理器的性能、功耗和功能。常见的指令集包括x86、ARM、MIPS等。

步骤：（注意： 不要传代码，而是传程序）

流程：

写代码（记得保存）main.c

编译： 用arm-linux-gcc来编译代码（在Ubuntu子系统中）

arm-linux-gcc main.c -o main

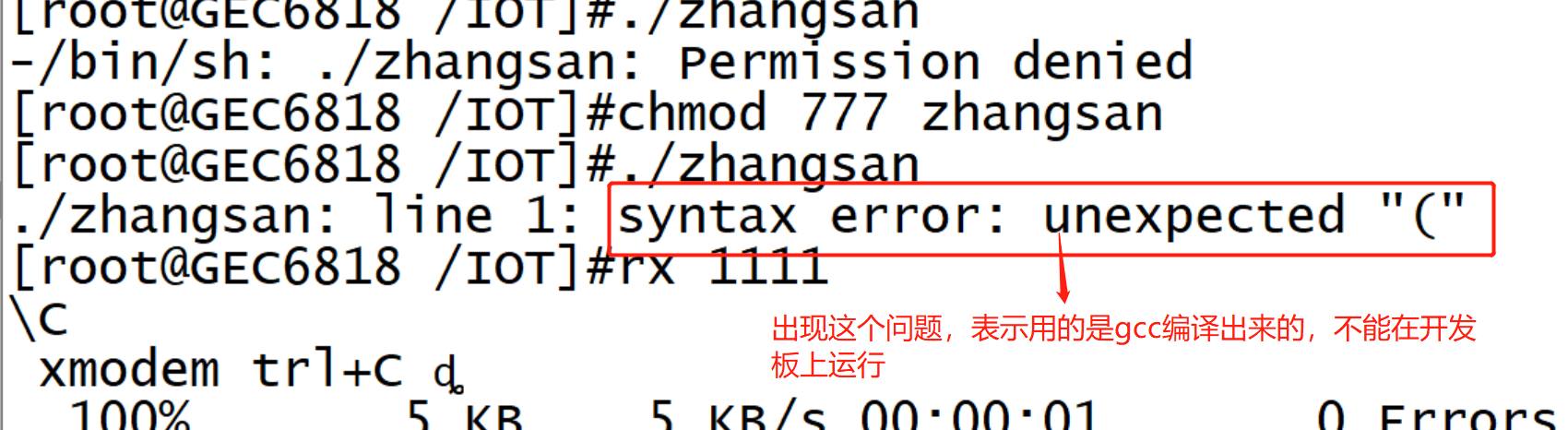
把编译生成的程序发送到开发板上

rx main

首次传输到开发板上的程序都是没有执行权限的，所以需要：

修改权限：chmod 777 main

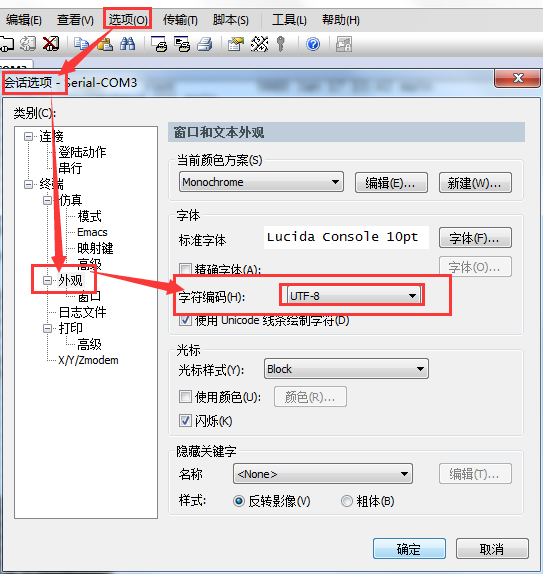
运行： ./main



如果中文出现乱码怎么解决

原因： 编码格式不对

修改编码格式为UTF-8



练习：

在开发板的终端上打印九九乘法表

提示：使用arm-linux-gcc工具包在Linux系统上编译

**网线传输文件(拓展)：**

流程：

先查看电脑的IP地址（1-3步）

修改开发板的IP地址（前三段要一致，最后一段不一样）（5步）

测试开发板跟电脑是否能正常通信（6步）

打开并设置tftp32.exe（7-8步）

传输文件（9步）

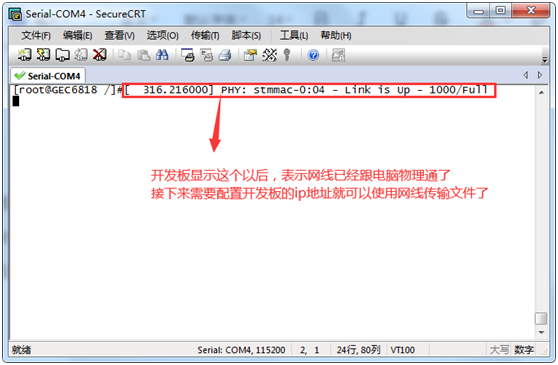
断开WiFi，接入网线，用网线链接电脑和开发板

电脑操作：win + R --》 输入cmd，呼出windows的命令窗口

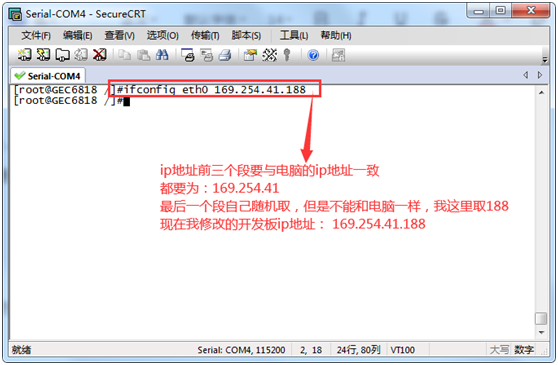
电脑操作： 在命令窗口内，输入ipconfig查询当前电脑的ip地址



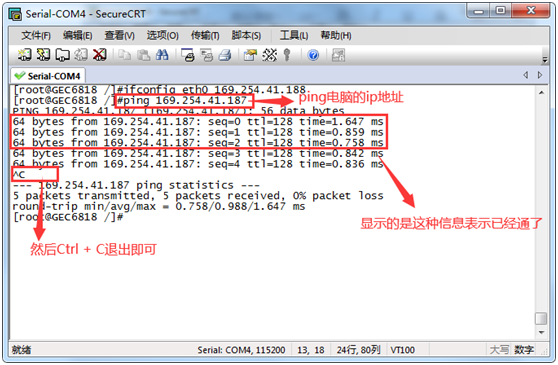
用网线连接电脑和开发板，开发板会显示：



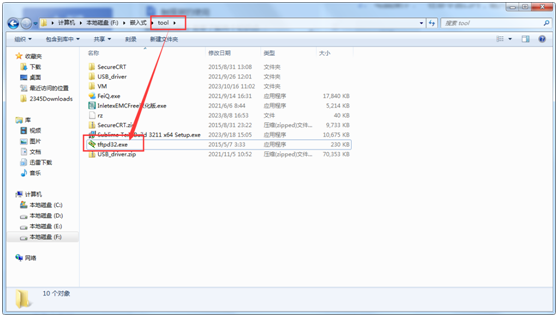
修改开发板的ip地址： ifconfig eth0 169.254.41.188



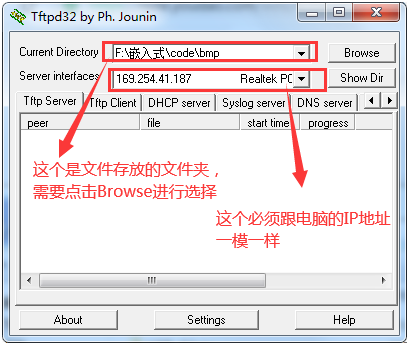
测试电脑是否可以和开发板ping通：



打开tftpd32.exe

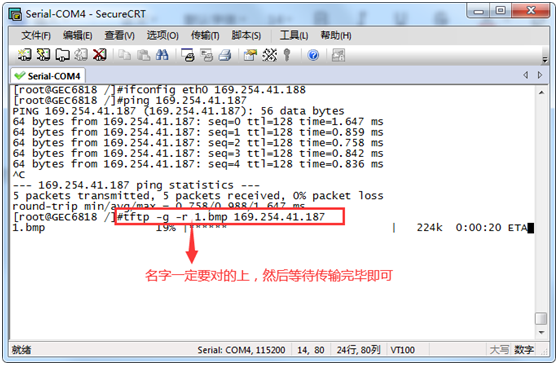


设置tftp32.exe



传输文件，命令： tftp -g -r +要传输的文件名 + 电脑的ip地址

例如：tftp -g -r 1.bmp 169.254.41.187



注意： 传输期间tftp32.exe程序不要关闭

**使用U盘进行传输（最快）**



注意：如果是exFat文件系统的U盘，是读不出来的

如果U盘是FAT32文件系统，它是直接可以使用的，不需要挂载

命令：

cd /mnt/udisk

ls// 就可以直接查看U盘的内容

如果U盘是NTFS文件系统，它是需要挂载才能使用

命令：

mount /dev/sda1 /mnt// 挂载U盘

cd /mnt// 切换

ls**// 查看U盘内**容

umount /dev/sda1// 不用记得卸载

注意：

先把文件拷贝到U盘里，把U盘插入开发板，最后使用cp命令把文件拷贝到开发板中

**LCD显示颜色**

目前市面上主流屏幕的种类： LCD、LED、OLED

深刻了解：在Linux内，一切皆为文件（在Linux诞生，是对的，随着Linux的不断发展，已经不太准确）

当我们需要使用lcd设备时，Linux已经把这个硬件设备抽象成了文件

LCD的设备文件：/dev/fb0

LCD设备文件的资源： 分辨率--》有多少个像素点，800\*480的分辨率，像素点：800\*480个像素点

像素点的组成：A（透明度）R（RED）G（GREEN）B（BLUE）---》4字节

LCD的设备文件大小： 800\*480\*4字节

代码如何实现？

我们日常生活里，怎么去保存数据的？打开--写入数据--保存退出

使用文件IO来完成这个功能，文件：硬件设备抽象出来的文件，IO：input输入output输出

代码：

#include <stdio.h>// #include: 预处理命令，包含所需的头文件

// stdio.h : 主要存放的是宏定义和函数的声明

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

int main(int argc, char const \*argv[])// main函数是程序的入口

{

// 1. 打开文件

int fd = open("/dev/fb0", O\_RDWR);

if (fd == -1)

{

printf("open file error\n");

return -1;

}

// 2. 写入数据（颜色）

int red\_color = 0xFF0000;

for (int i = 0; i < 800\*480; ++i)

{

write(fd, &red\_color, 4);

}

// 3. 保存退出（自动保存的功能）

int ret = close(fd);

if (ret == -1)

{

printf("close file error\n");

return -2;

}

return 0;// 程序的出口，只要程序执行到这里，便会无条件结束

}

lcd设备文件的使用问题：

为什么一个循环，可以挨着写入数据？

练习：开发板显示德国国旗



拓展： 开发板显示法国国旗



//法国国旗

for(y=0; y<480; y++)// ys

{

for(x=0; x<800; x++)//x

{

if(x<800/3)

{

write(fd, &blue\_color, 4);

}

else if (x>800/3 && x<800/3\*2)

{

write(fd, &wirte\_color, 4);

}

else

{

write(fd, &red\_color, 4);

}

}

}