**江苏科技大学**

**课 设 报 告**

课 程： 嵌入式系统与安全

项 目： 嵌入式加密通信系统

学 院： 计算机学院

专 业： 信息安全

班 级： 1822107101

组 员： 182210710119 陈四贵

182210710140 赵富星

182210710113 万文慧

指导老师： 江苏科技大学 张 明

目 录

[一、设计目的 2](#_Toc89109756)

[二、设计原理 2](#_Toc89109757)

[三、设计步骤、记录与分析 11](#_Toc89109758)

[四、设计小结 19](#_Toc89109759)

# 一、设计目的

1、通过嵌入式项目设计，掌握嵌入式开发的原理、方法和应用；

2、掌握较复杂的加密算法，并将其应用至系统通信中；

3、掌握嵌入式系统中的串口通信技术。

# 二、设计原理

1、嵌入式系统开发流程

嵌入式linux开发，根据应用需求的不同有不同的配置开发方法，但是- -般都要经过以下过程:

①建立开发环境。 操作系使用Ubuntu14.04。

②配置开发主机.配置串口终端软件参数为波特率115200，数据位8位，停止位1,无奇偶校验，软硬件控制流设为无。在WINDOWS下的超级终端的配置是这样。MINICOM问超级终端软件的作用一样是作为调试嵌入式开发板信息输出的监视器和键盘输入的工具:配置网络.主要是配置NFS网络文件系统，需要关闭防火墙，简化嵌入式网络调试环境设置过程。配置SMB服务来实现Windows系统与LINUX系统文件共享。建立引导装载程序BOOTLOADER,从网络上下载- -些公开源代码的BOOTLOADER,如U-BOOT、BLOB、VIVI、 LILO、 ARM-BOOT. RED-BOOT等，根据自己具体芯片进行移植修改。有些芯片没有内置引导装载程序，比如三星的ARM7. ARM9 系列芯片，这样就需要编写烧写开发板上fash的烧写程序，网络上有免费下载的WINDOWS下通过JTAG并口简易仿真器烧写ARM外围flash芯片的程序。也有LINUX下公开源代码的JFLASH程序。如果不能烧写自己的开发板，就需要根据自己的具体电路进行源代码修改。这是让系统可以正常运行的第-一步。 如果你购买了厂家的仿真器当然比较容易烧写flash了,但是其中的核心技术是无法了解的。这对于需要迅速开发应用的人来说可以极大地提高开发速度。

③下载别人已经移植好的 LINUX操作系统，如UCLINUX. ARM-LINUX. PPC-LINUX等，如果

有专门针对你所使用的CPU移植好的LINUX操作系统那是再好不过，下载后再添加自己的特定硬件的驱动程序，进行调试修改，对于带MMU的CPU可以使用模块方式调试驱动，对于UCLINUX这样的系统只能编译进内核进行调试。

④建立根文件系统， 从www.busybox. net下载使用BUSYBOX软件进行功能裁减，产生一个最基本的根文件系统，再根据自己的应用需要添加其他的程序。默认的启动脚本- - 般都不会符合应用的需要，所以就要修改根文件系统中的启动脚本，它的存放位置位于etc目录下，包括: /tchnit.drc.S. /etclprofle/etc/.profile等，自动挂装文件系统的配置文件letc/fstab，具体情况会随系统不同而不同。根文件系统在嵌入式系统中- -般设为只读，需要使用mkcramt's . genronfs 等工具产生烧写映象文件。

⑤建立应用程序的 flash磁盘分区，一般使用JFFS2或YAFFS文件系统，这需要在内核中提供这些文件系统的驱动，有的系统使用一个线性flash (NOR型) 512K- 32M, 有的系统使用非线性flash (NAND型) 8-512M.有的两个同时使用，需要根据应用规划flash的分区方案。

⑥开发应用程序，可以下载到根文件系统中，也可以放入YAFFS. JFPS2文件系统中，有的应用程序不使用根文件系统，而是直接将应用程序和内核设计在- -起， 这有点类似于UCOS-II的方式。

⑦烧写内核、根文件系统、应用程序。最终发布产品。

2、Linux系统操作

Linux系统通常使用终端进行操作，常见操作命令如下：

①权限授予类

|  |
| --- |
| 授予最高级管理员权限：sudo su  收回管理员权限：su <普通用户名> |

②目录跳转类

|  |
| --- |
| 查看当前目录所有文件：ls  查看当前目录所有文件（包括隐藏文件）：ls -a  创建目录：mkdir <目录名>  删除空目录：rmdir <目录名>  强行删除整个目录内容（无法恢复）：rm -rf  进入当前目录的某一子目录：cd <子目录名>  返回上一级目录：cd ..  显示当前目录路径：pwd  列举当前 TTY 下所有进程：ps  列举所有：ps -A  临时修改 PC 机 IP 地址：ifconfig eth0 192.168.X.XXX  在一个目录树中查找含有某个字符串的所有文件：grep ‘Modified by zou’ \* -r |

③文件操作类（以Vim、gedit为例）

|  |
| --- |
| vim：  创建/打开一个txt文件：vim <文件名>  保存并退出：Esc :wq!  gedit：  打开一个文件：gedit <文件名> |

④网络配置类

|  |
| --- |
| 查看当前网络配置：ipconfig  配置端口地址：ipconfig eth0 192.168.80.111 netmask 255.255.255.0 |

3、嵌入式加密通信系统开发流程

①设计加密算法

②与底层通信算法以符合我们的需要

③配置串口，并通过串口传送数据。

4、数据加密算法·Vigenere 算法

|  |
| --- |
| //加密  int encrypt(char \*text,char \*result,char \*k)  {  int i,j=0,z=0;  int m = strlen(k); //获取密钥的长度  int l = strlen(text); //获取明文的长度  for(i=0;i<l;i++)  {  //判断大小写  if (text[i] >= 'A' && text[i] <= 'Z'){  if(j==m){  j=0; //循环密钥  result[z]=(text[i]-'A'+k[j]-'A')%26+'A';  } else {  result[z]=(text[i]-'A'+k[j]-'A')%26+'A';  }  j++;  } else if (text[i] >= 'a' && text[i] <= 'z'){  if(j==m){  j=0; //循环密钥  result[z]=(text[i]-'a'+k[j]-'a')%26+'a';  } else {  result[z]=(text[i]-'a'+k[j]-'a')%26+'a';  }  j++;  } else{ //判断是否是空格  result[z] = text[i];  }  z++;  }  return 0;  }  //解密  int decrypt(char \*text,char \*result,char \*k)  {  int i,j=0,z=0;  int m = strlen(k); //获取密钥的长度  int l = strlen(text); //获取密文的长度  for(i=0;i<l;i++)  {  //判断是否是空格  if (text[i] >= 'A' && text[i] <= 'Z'){  if(j==m){  j=0; //循环密钥  result[z]=(text[i]-k[j]+26)%26+'A';  } else {  result[z]=(text[i]-k[j]+26)%26+'A';  }  j++;  } else if (text[i] >= 'a' && text[i] <= 'z'){  if(j==m){  j=0; //循环密钥  result[z]=(text[i]-k[j]+26)%26+'a';  } else {  result[z]=(text[i]-k[j]+26)%26+'a';  }  j++;  } else{  result[z] = text[i];  }  z++;  }  return 0;  } |

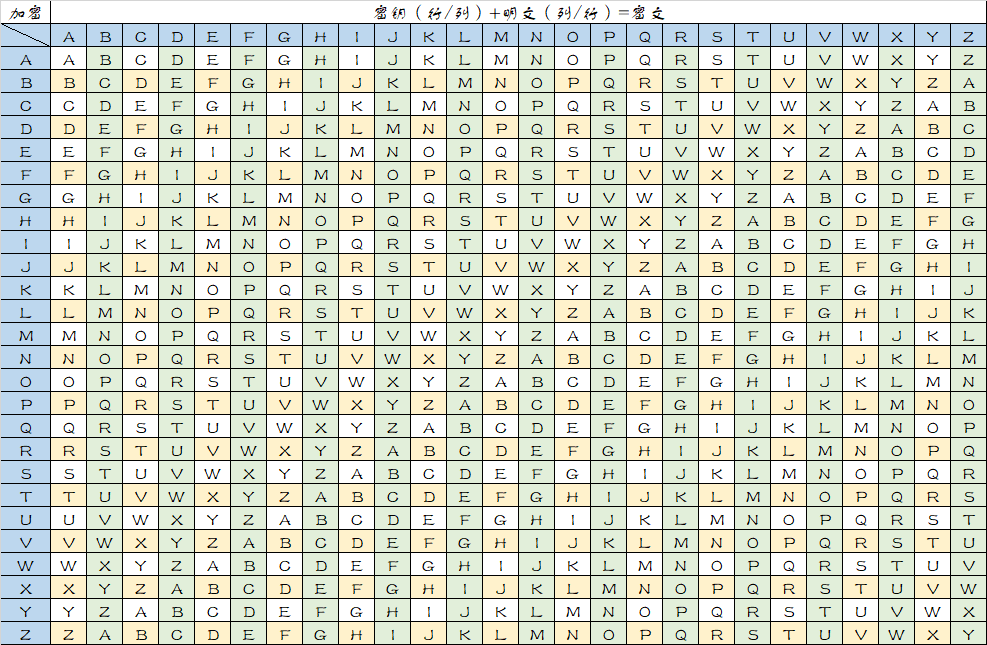


图1 vigenere密码加密示意图

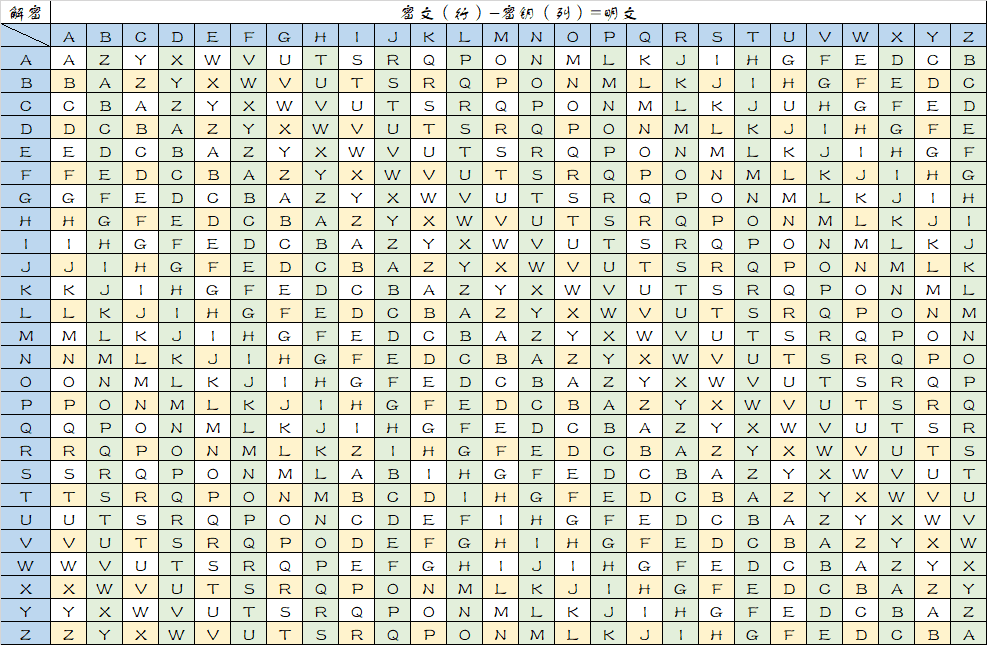


图2 vigenere密码解密示意图

5、嵌入式系统中的界面设计·

①Qt与Qt Creator简介：Qt是一个跨平台应用程序和 UI 开发框架。使用 Qt 您只需一次性开发应用程序，无须重新编写源代码，便可跨不同桌面和嵌入式操作系统部署这些应用程序。Qt Creator 是全新的跨平台Qt IDE，可单独使用，也可与 Qt 库和开发工具组成 一套完整的SDK. 其中包括：高级 C++ 代码编辑器，项目和生成管理工具，集成的上下文相关的帮助系统，图形化调试器，代码管理和浏览工具。

②Qt功能与特性：

◎直观的 C++ 类库：模块化 Qt C++ 类库提供一套丰富的应用程序生成块 (block)，包含了构建高级跨平台应用程序所需的全部功能。具有直观，易学、易用，生成好理解、易维护的代码等特点。

◎跨桌面和嵌入式操作系统的移植性：使用 Qt，您只需一次性开发应用程序，就可跨不同桌面和嵌入式操作系统进行部署，而无须重新编写源代码，可以说Qt无处不在（QtEverywhere） 。

◎使用单一的源代码库定位多个操作系统；

◎通过重新利用代码可将代码跨设备进行部署；

◎无须考虑平台，可重新分配开发资源；

◎代码不受担忧平台更改影响的长远考虑 ；

◎使开发人员专注于构建软件的核心价值，而不是维护 API 。

◎具有跨平台 IDE 的集成开发工具：Qt Creator 是专为满足 Qt 开发人员需求而量身定制的跨平台集成开发环境 (IDE)。Qt Creator 可在 Windows、Linux/X11 和 Mac OS X 桌面操作系统上运行，供开发人员针对多个桌面和移动设备平台创建应用程序。

在嵌入式系统上的高运行时间性能，占用资源少。

③Qt Creator功能和特性

◎复杂代码编辑器：Qt Creator 的高级代码编辑器支持编辑 C++ 和 QML (JavaScript)、上下文相关帮助、代码完成功能、本机代码转化及其他功能。

◎版本控制：Qt Creator 汇集了最流行的版本控制系统，包括 Git、Subversion、Perforce、CVS 和 Mercurial。

◎集成用户界面设计器：Qt Creator 提供了两个集成的可视化编辑器：用于通过 Qt widget 生成用户界面的 Qt Designer，以及用于通过 QML 语言开发动态用户界面的 Qt Quick Designer\*。

◎项目和编译管理 ：无论是导入现有项目还是创建一个全新项目，Qt Creator 都能生成所有必要的文件。包括对 cross-qmake 和 Cmake 的支持。

◎桌面和移动平台：Qt Creator 支持在桌面系统和移动设备中编译和运行 Qt 应用程序。通过编译设置您可以在目标平台之间快速切换。

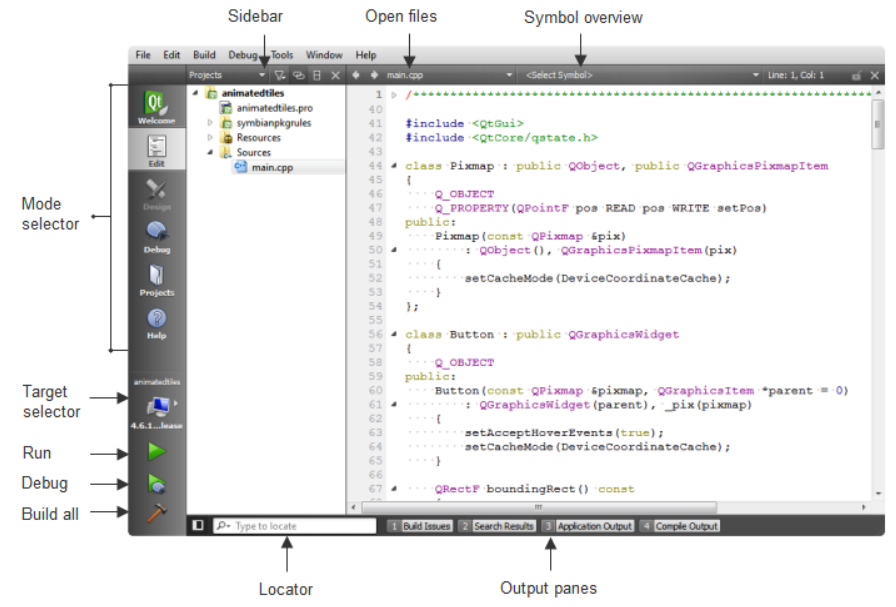
◎Qt 模拟器：Qt模拟器是诺基亚 Qt SDK的一部分，可在与目标移动设备相似的环境中对移动设备的 Qt 应用程序进行测试。

④Qt类库

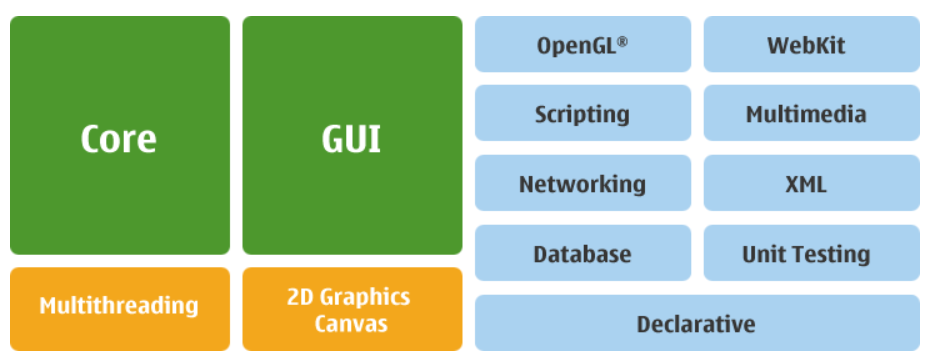
◎模块化 Qt C++ 类库提供一套丰富的应用程序生成块(block)，包含了生成高级跨平台应用程序所需的全部功能。

◎先进的图形用户界面（GUI）：Qt为您在桌面与嵌入式平台上开发先进的GUI应用程序，带来所有需要的功能。Qt使用所支持平台的本地化图形API，充分利用系统资源并给予应用程序本地化的界面。

◎从按钮和对话框到树形视图与表格都具有完整的控件（窗体）自动缩放，字体、语言与屏幕定位识别布局引擎支持抗锯齿、矢量变形以及可缩放矢量图形 (SVG)具有样式API和窗体样式表，可完全自定义用户界面支持嵌入式设备的硬件加速图形和多重显示功能

◎基于OpenGL ®与OpenGL ®Es的3D图形：OpenGL® 是一个标准的图形库，用于构建跨平台和支持硬件加速的高性能可视化应用程序。虽然OpenGL完美支持3D图形，但却不支持创建应用程序用户界面。Qt通过与OpenGL 的紧密集成解决了这一难题。

◎多线程：多线程编程是一个执行资源密集型操作而不会冻结应用程序用户界面的有效典范。Qt的跨平台多线程功能简化了并行编程，另外它附加的同步功能可以更加轻松地利用多核架构。管理线程、数据和对象更加轻松。基于Qt的信号与槽，实现跨线程类型安全的对象间通讯高端API可以编译多线程程序而无须使用底端基元。

◎嵌入式设备的紧凑视窗系统：Qt构建在标准的 API基础上，用于具有轻量级window系统的嵌入式 Linux 设备。基于 Qt的应用程序直接写入 Linux 帧缓冲，解除了您对 X11 视窗系统的需求。减少内存消耗，内存占用更小；可以编译移除不常使用的组件与功能；可以利用硬件加速图形；在桌面系统上的虚拟帧缓冲可用于嵌入式开发与调试

⑤对象间通讯：在开发用户图形界面中，一个常见的、重复发生系统崩溃与问题的症结根源是如何在不同组件之间进行通信。对于该问题，Qt 的解决方案是信号与槽机制，即执行Observer设计模式。我们可以简单理解为当特殊事件发生的时候，信号就被发出了，一个插槽就是一个函数，被称作特定信号的响应。

6、串口通信原理与实现

|  |
| --- |
| #include <termios.h>  #include <stdio.h>  #include <termios.h>  #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <fcntl.h>  #include <sys/types.h>  #include <string.h>  #include <sys/signal.h>  #include <pthread.h>  #define BAUDRATE B115200  #define COM "/dev/ttymxc4" //串口1设备  #define FALSE 0  #define TRUE 1  static int fd;  pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;  void com\_init(speed\_t speed)  {  struct termios options;  fd = open(COM, O\_RDWR | O\_NOCTTY | O\_NDELAY);// | O\_NONBLOCK  if(fd < 0)  {  printf("open com device failure");  }  tcgetattr(fd,&options);  cfsetispeed(&options,speed);//波特率  cfsetospeed(&options,speed);  options.c\_cflag |= (CLOCAL|CREAD);  options.c\_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);  options.c\_oflag &= ~OPOST;  options.c\_iflag &= ~(BRKINT | ICRNL | INPCK | ISTRIP | IXON);  tcsetattr(fd,TCSANOW,&options);  }  /\*--------------------------------------------------------\*/  //加密  int encrypt(char \*text,char \*result,char \*k)  {  int i,j=0,z=0;  int m = strlen(k); //获取密钥的长度  int l = strlen(text); //获取明文的长度  for(i=0;i<l;i++)  {  //判断大小写  if (text[i] >= 'A' && text[i] <= 'Z'){  if(j==m){  j=0; //循环密钥  result[z]=(text[i]-'A'+k[j]-'A')%26+'A';  } else {  result[z]=(text[i]-'A'+k[j]-'A')%26+'A';  }  j++;  } else if (text[i] >= 'a' && text[i] <= 'z'){  if(j==m){  j=0; //循环密钥  result[z]=(text[i]-'a'+k[j]-'a')%26+'a';  } else {  result[z]=(text[i]-'a'+k[j]-'a')%26+'a';  }  j++;  } else{ //判断是否是空格  result[z] = text[i];  }  z++;  }  return 0;  }  //解密  int decrypt(char \*text,char \*result,char \*k)  {  int i,j=0,z=0;  int m = strlen(k); //获取密钥的长度  int l = strlen(text); //获取密文的长度  for(i=0;i<l;i++)  {  //判断是否是空格  if (text[i] >= 'A' && text[i] <= 'Z'){  if(j==m){  j=0; //循环密钥  result[z]=(text[i]-k[j]+26)%26+'A';  } else {  result[z]=(text[i]-k[j]+26)%26+'A';  }  j++;  } else if (text[i] >= 'a' && text[i] <= 'z'){  if(j==m){  j=0; //循环密钥  result[z]=(text[i]-k[j]+26)%26+'a';  } else {  result[z]=(text[i]-k[j]+26)%26+'a';  }  j++;  } else{  result[z] = text[i];  }  z++;  }  return 0;  }  /\* modem input handler \*/  void\* receive(void \* data)  {  char ch[1024];  int ret;  printf("read modem\r\n");  while (1)  {  ret = read(fd,ch,sizeof(ch)); //读串口0数据  if(ret > 0)  {  ch[ret] = 0;  //printf("c=%s\n",ch);  pthread\_mutex\_lock(&mutex);  write(fd,ch,strlen(ch)); //将读到的数据输出到串口0上  pthread\_mutex\_unlock(&mutex);  }  }  printf("\r\n");  printf("exit from reading modem\n");  return NULL;  }  /\*--------------------------------------------------------\*/  void\* send(void \* data)  {  printf("send data\r\n");  char text[50]="";  char result[50]="";  char k[50]="";  char res[1024];  printf("请输入明文:\n");  gets(text);  printf("请输入秘钥key:\n");  gets(k);  encrypt(text,result,k);  printf("明文%s的密文为:%s\n",text,result);  decrypt(text,result,k);  printf("密文%s的明文为:%s\n",result,text);  pthread\_mutex\_lock(&mutex);  int lens = strlen(result); //获取的长度  write(fd,result,lens); //循环发送字符0~9到串口0上  pthread\_mutex\_unlock(&mutex);  return NULL; /\* wait for child to die or it will become a zombie \*/  }  /\*--------------------------------------------------------\*/  int main(int argc,char\*\* argv)  {  pthread\_t th\_a, th\_b;  void \* retval;    pthread\_mutex\_init(&mutex,NULL);  com\_init(BAUDRATE);  pthread\_create(&th\_b, NULL, send, 0);  pthread\_create(&th\_a, NULL, receive, 0);    //等待子线程结束  pthread\_join(th\_a, &retval);  pthread\_join(th\_b, &retval);  close(fd);  exit(0);  } |

用线连接其它开发板并运行程序，或者连接 485 转 232 到上位机并打开串口调试助手。在 arm 串口上输入信息回车，在其它开发板或上位机上会显示相应信息，同时可以回复信息并在串口上显示出来。

示例：在两台设备的串口终端分别执行程序， 其中一端输入 data 之后， 在另外的一端就会有对应数据的打印，打印的数据会在 The information is 后面区域显示。

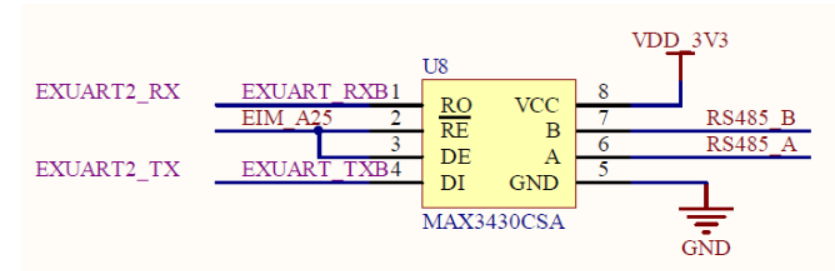
注意：两个平台互相发数据需要来发切换串口终端进行操作。在一端输入完一段数据之后，在另外一段这段数据就会显示。此时在有数据显示的串口终端上面进行操作，继续输入数据即可

图3 原理图

# 三、设计步骤、记录与分析

①配置主机与虚拟机网络，调整虚拟机的网络模式为桥接，使之联通。

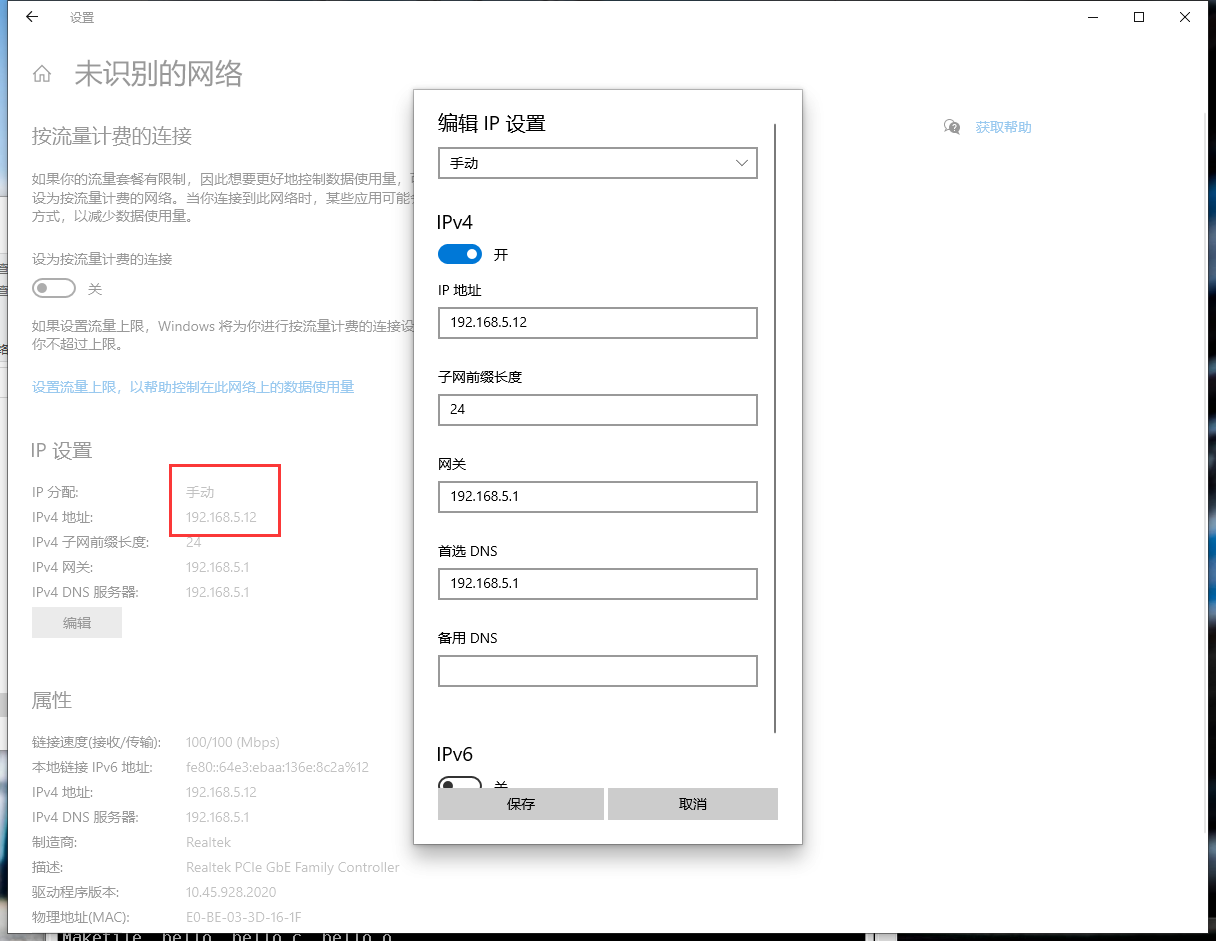


图1-1 配置主机网络IP地址为192.168.5.12

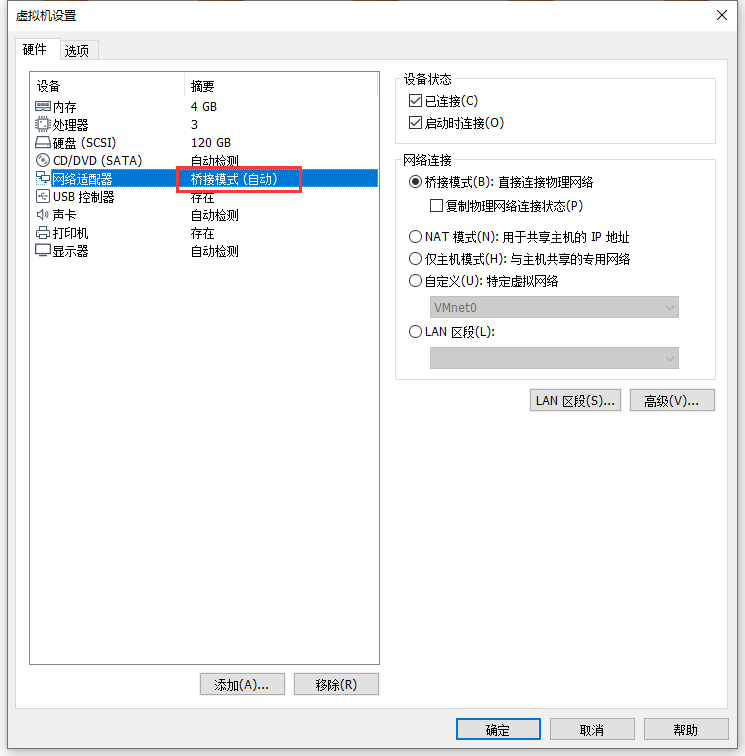


图1-2 设置虚拟机网络模式为桥接模式

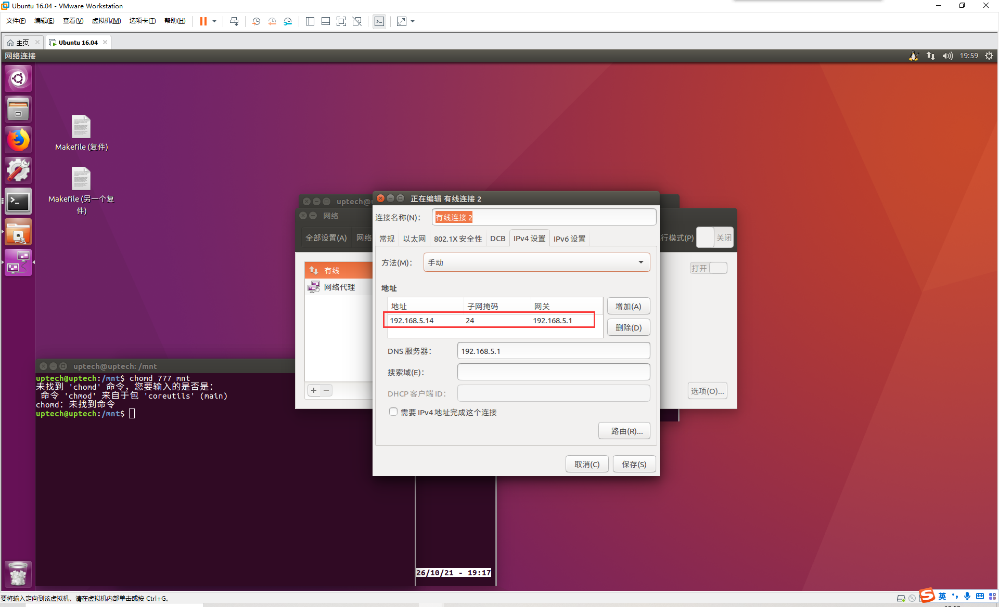


图1-3 设置虚拟机网络相关参数（地址、子网掩码、网关等）

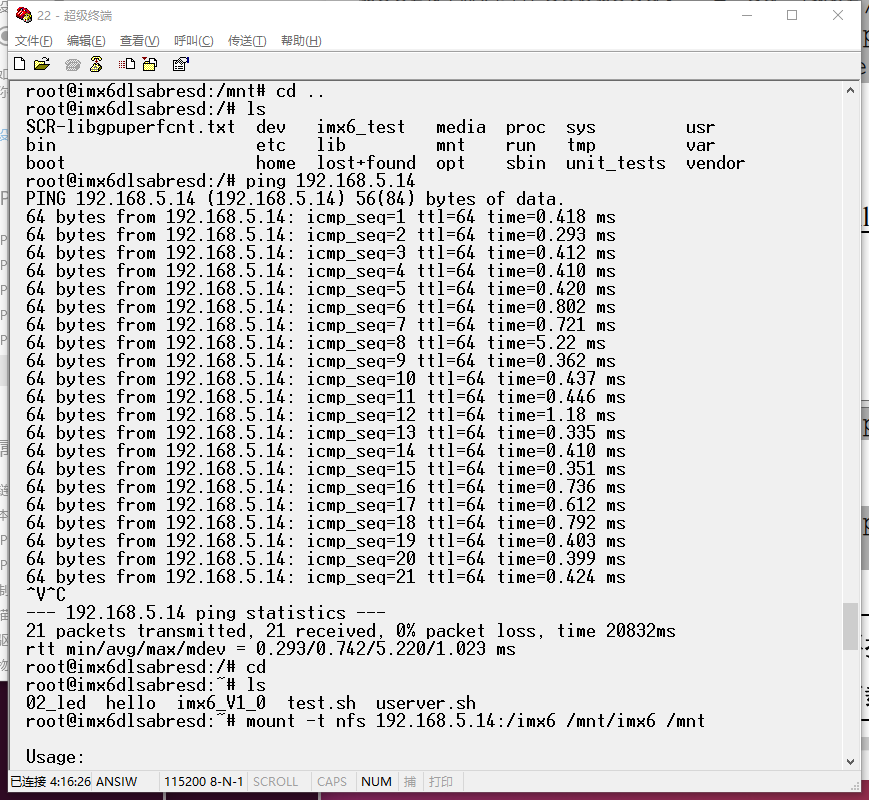


图1-4 在主机中执行Ping命令，测试是否联通。结果显示是联通的。

②在虚拟机中编写Hello world程序，并将其放置共享文件目录

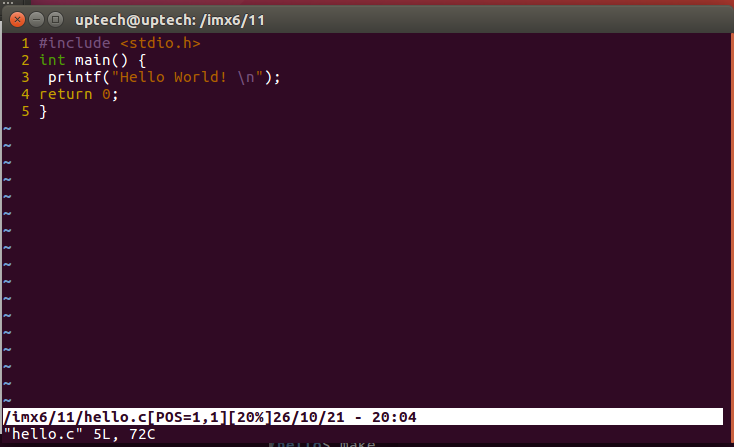


图2-1 创建“hello.c”文件，并通过vim编辑器编写程序

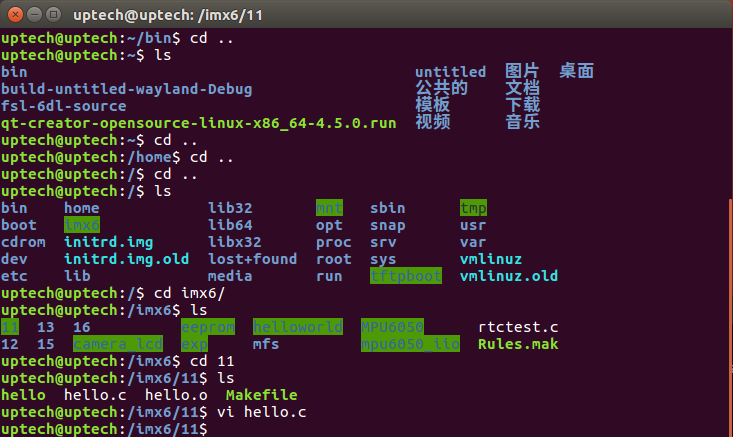


图2-2 通过ls命令查看当前目录的文件

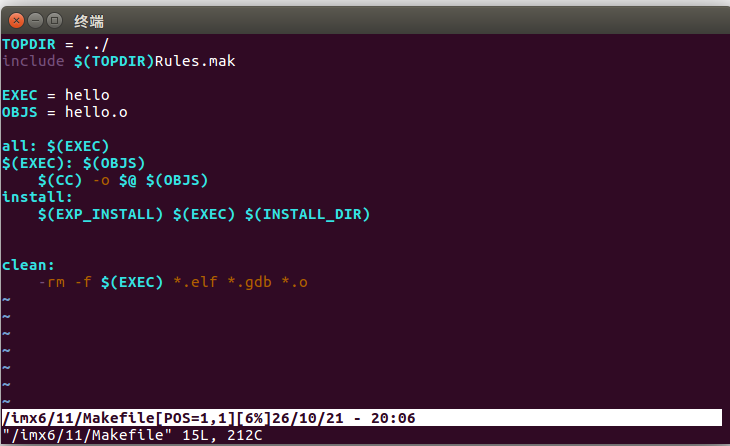


图2-3 由于新安装虚拟机环境的缺陷，此处需配置虚拟机环境参数

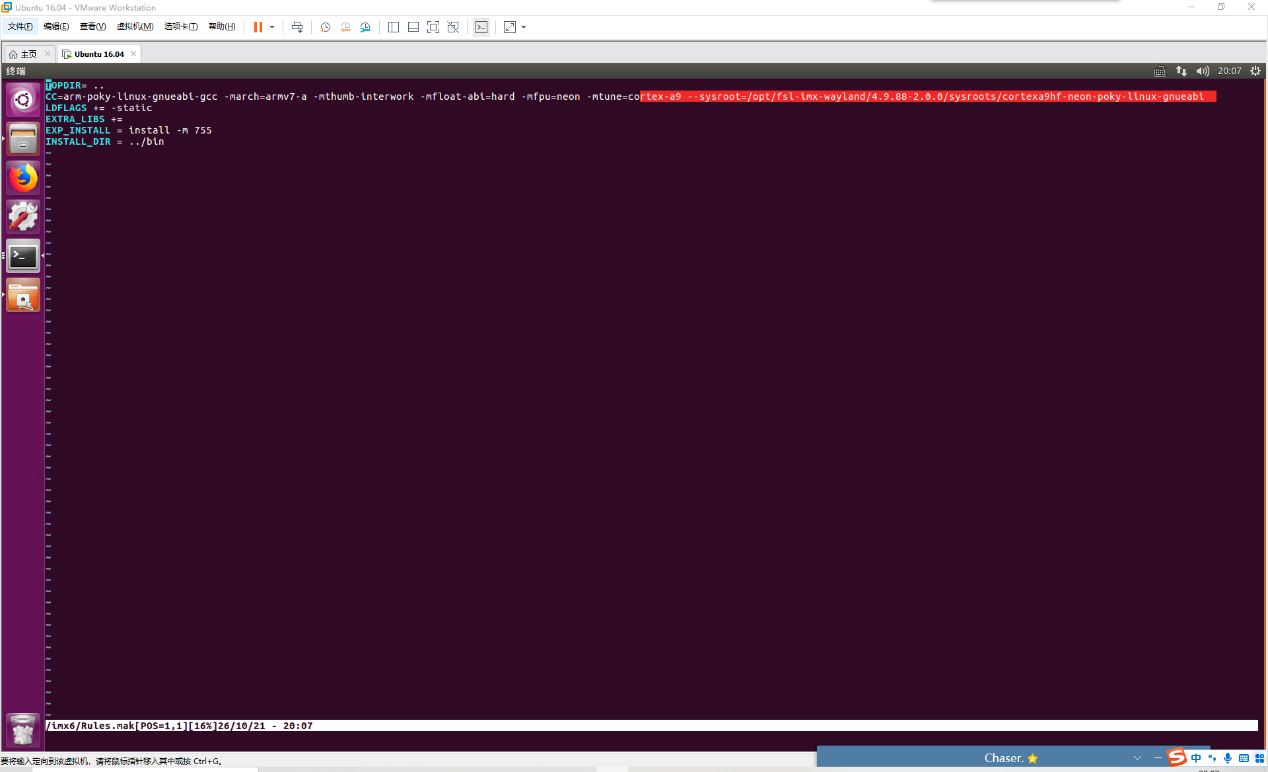


图2-4 此处显示，环境参数配置成功

③把003\_tty文件夹复制到imx6文件夹下

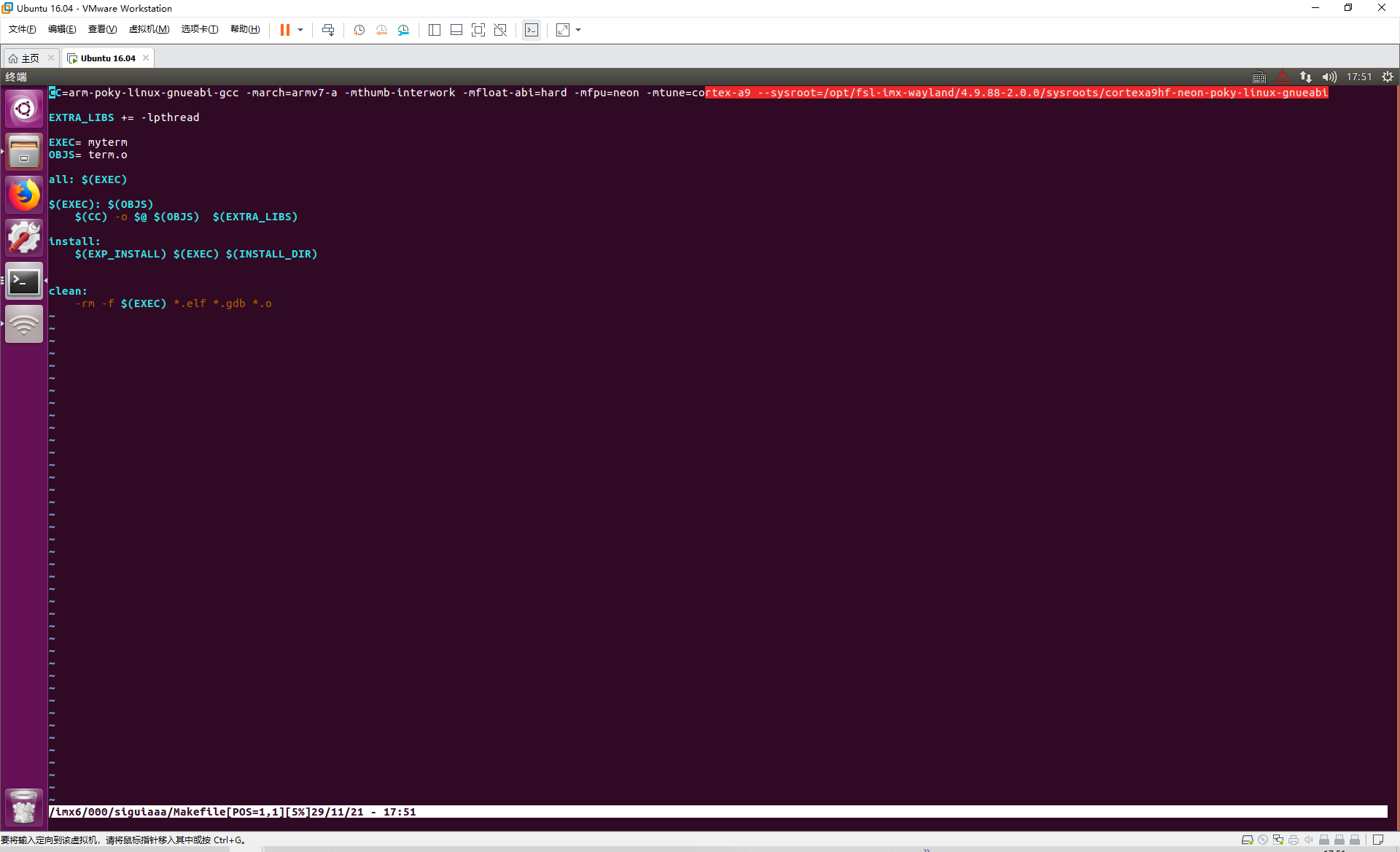


图3-1 示意图

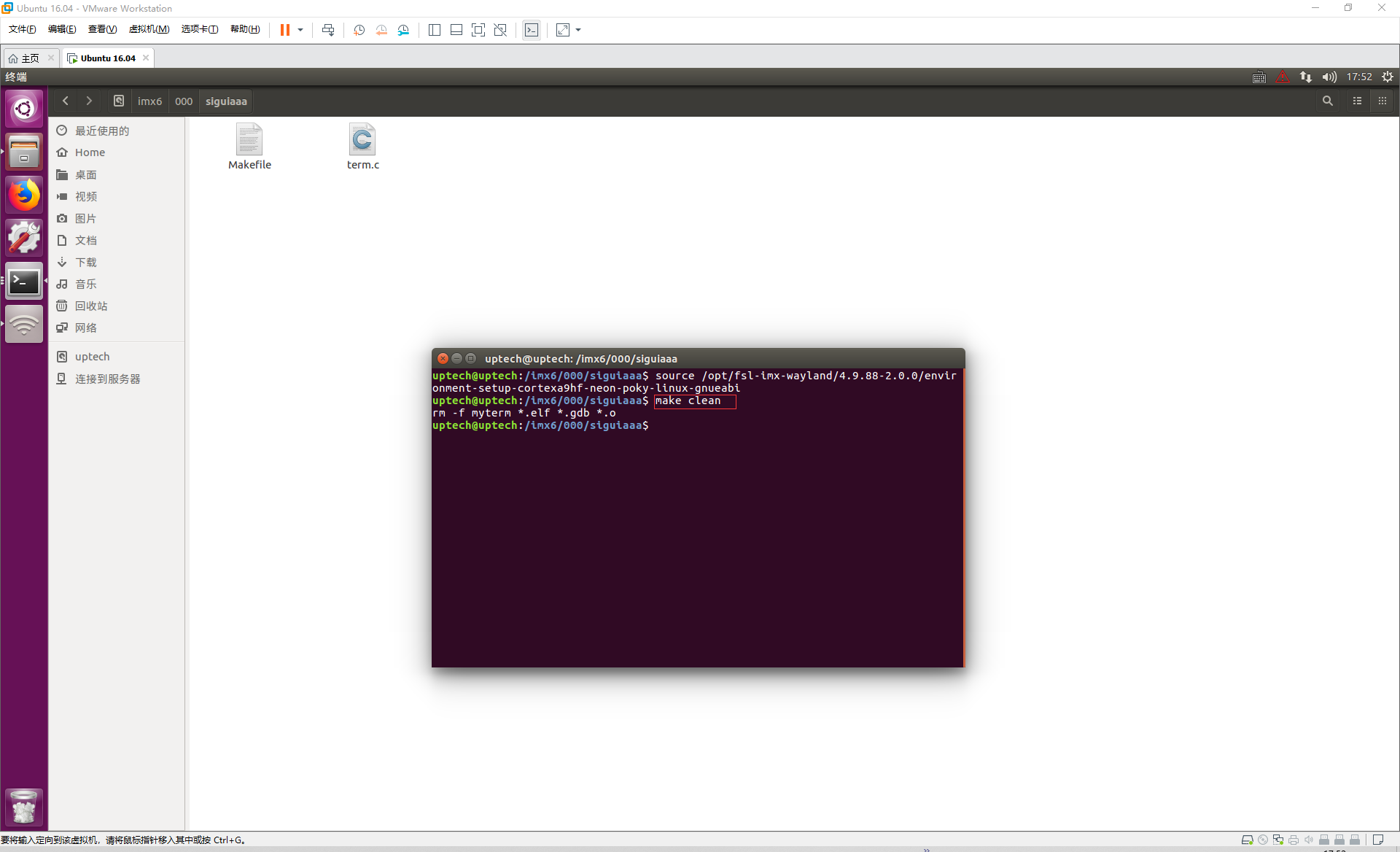


图3-2 示意图

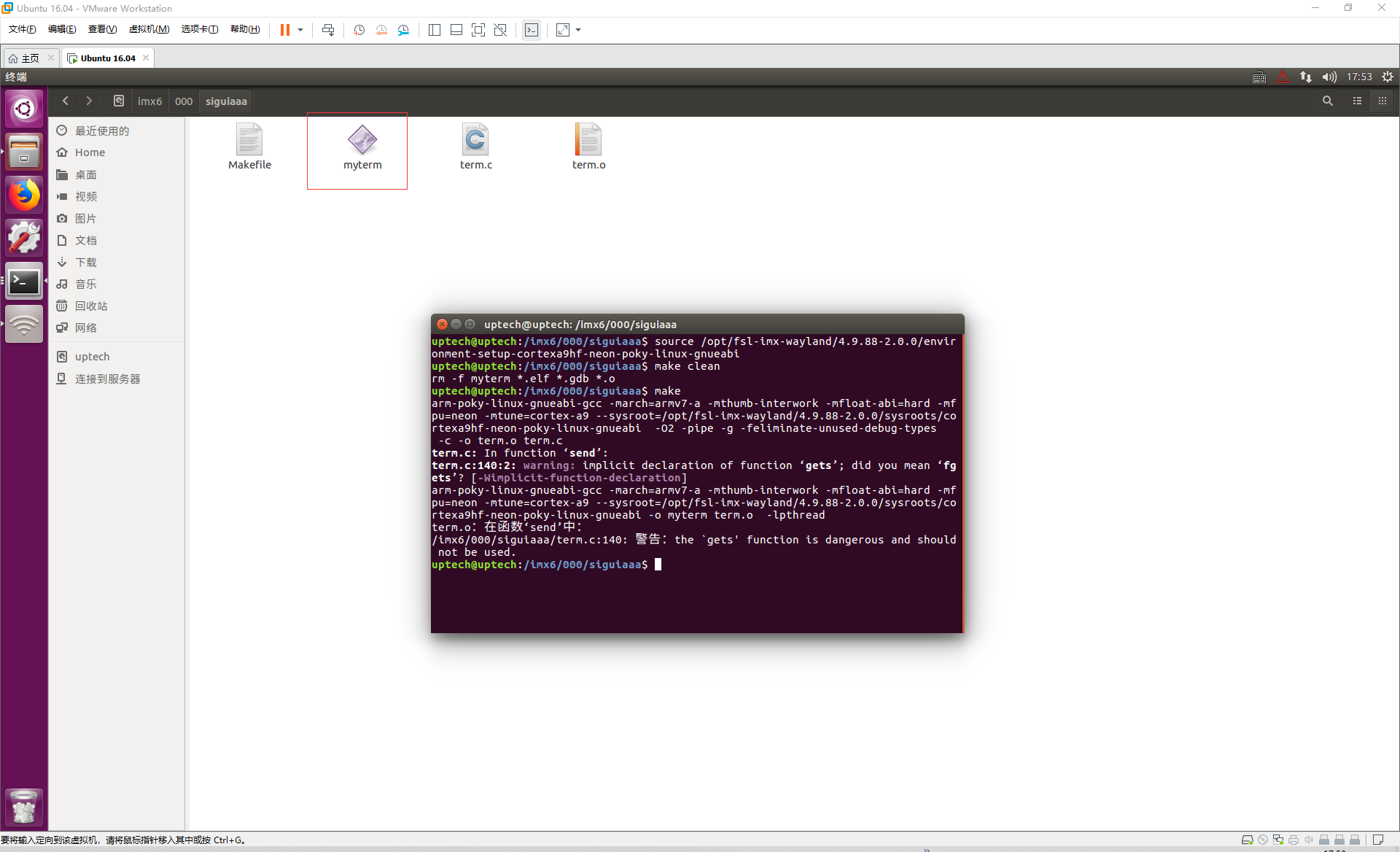


图3-3 示意图

④挂载至mnt目录

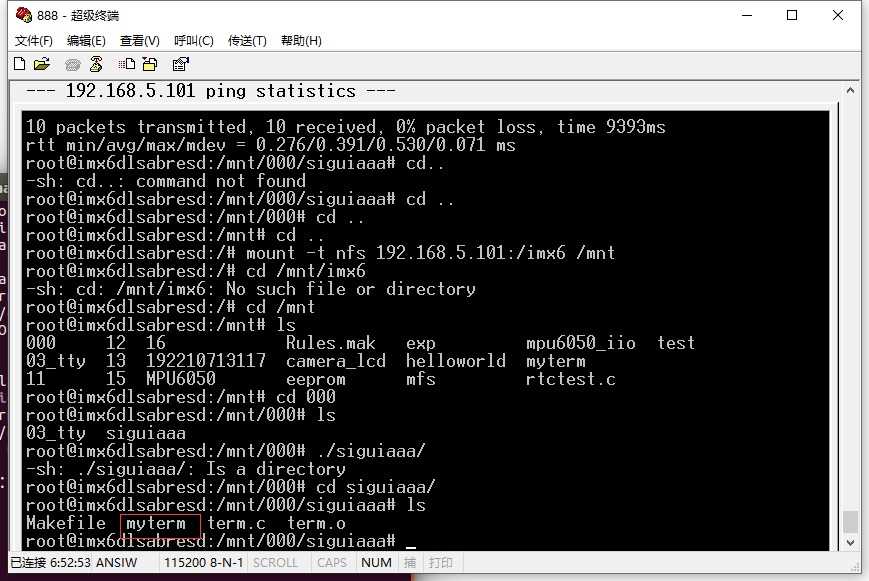


图4-1 挂载示意图

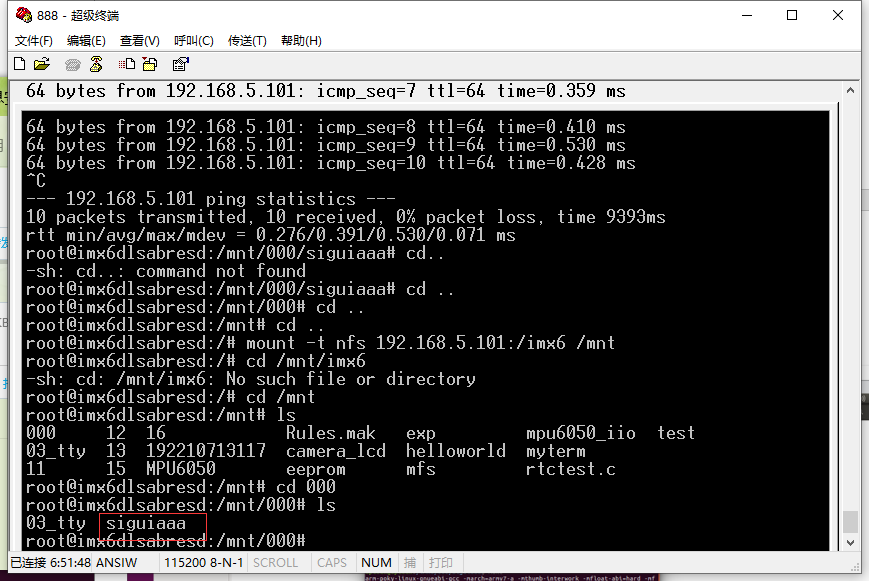


图4-2 挂载示意图

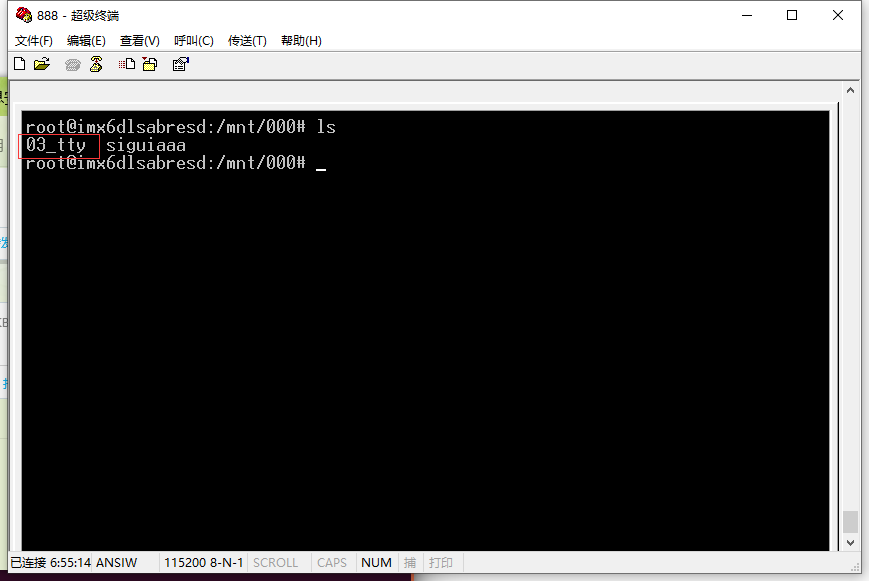


图4-3 挂载示意图

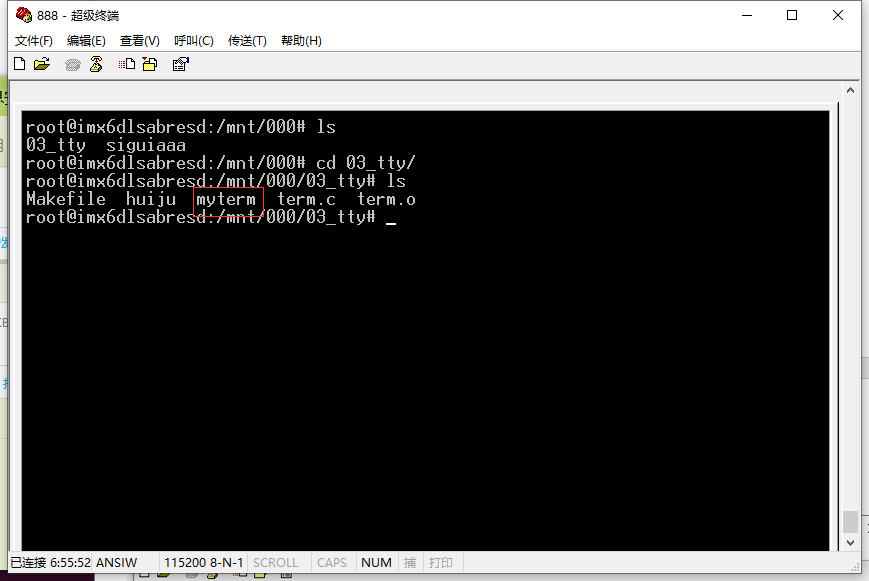


图4-4 挂载示意图

⑤加密

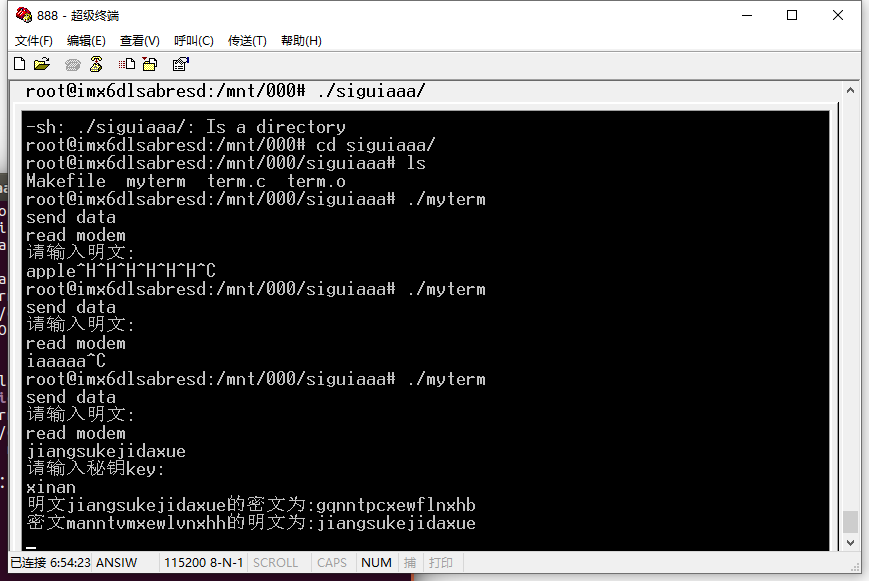


图5-1 加密示意图

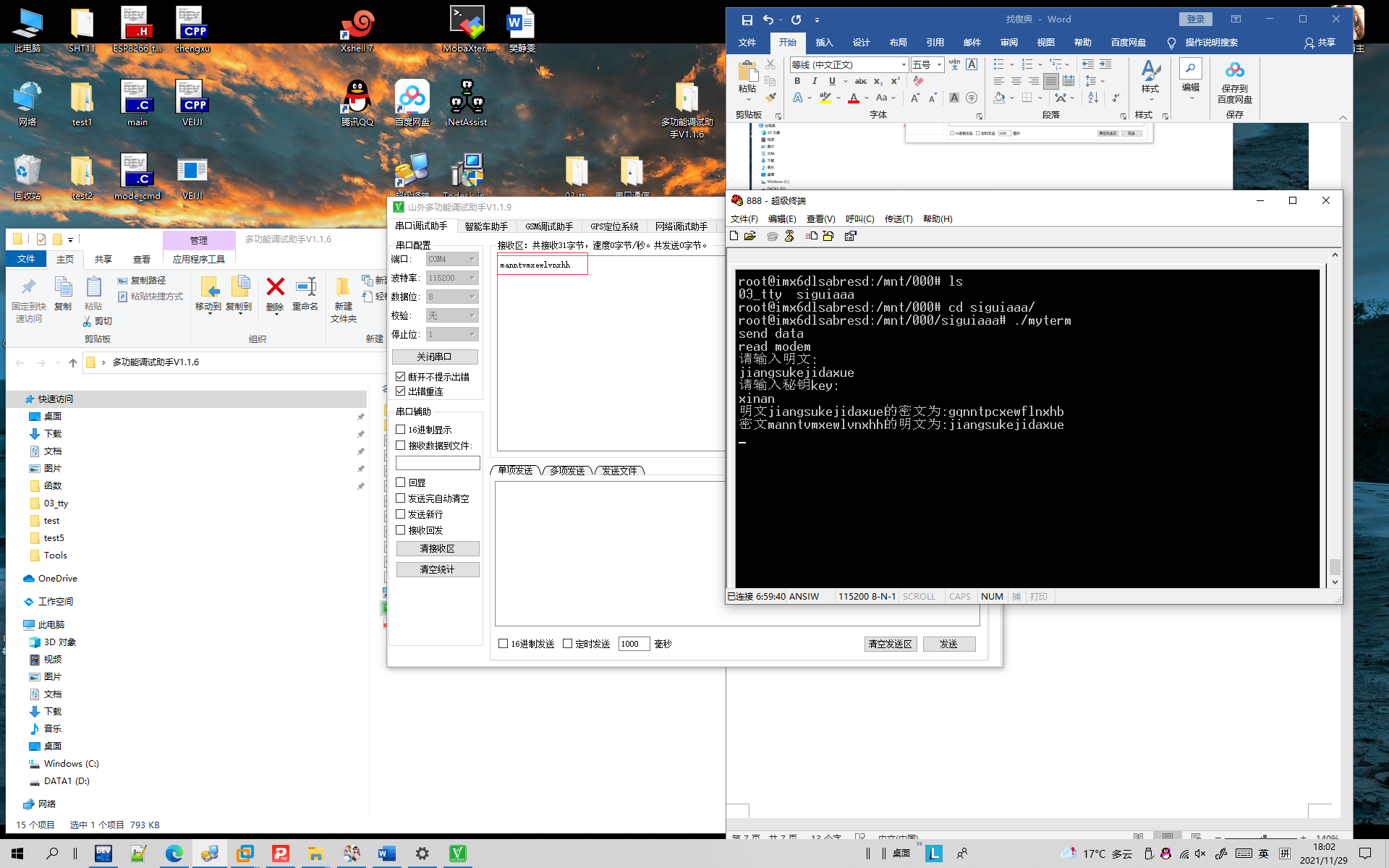


图5-2 通信示意图

# 四、设计小结

1、关于网络联通：vmware中有三种网络连接方式——NAT、Bridged Adapter与Host-only Adapter，它们的通信范围各不相同，如下图所示。

|  |  |  |  |
| --- | --- | --- | --- |
|  | NAT | Bridged Adapter | Host-only Adapter |
| 虚拟机->主机 | √ | √ | 默认不能，需设置 |
| 主机->虚拟机 | × | √ | 默认不能，需设置 |
| 虚拟机->其他主机 | √ | √ | 默认不能，需设置 |
| 其他主机->虚拟机 | × | √ | 默认不能，需设置 |
| 虚拟机之间 | × | √ | √ |

2、关于设备挂载：挂载（mounting）是指由操作系统使一个存储设备（通常为共享资源）上的计算机文件和目录可供用户通过计算机的文件系统访问的一个过程。一般来说，当计算机关机时，每个已挂载存储都将经历一次卸载，以确保所有排队的数据被写入，并保证介质上文件系统结构的完整性。

3、关于串口通信：串行接口是一种可以将接收来自CPU的并行数据字符转换为连续的串行数据流发送出去，同时可将接收的串行数据流转换为并行的数据字符供给CPU的器件。在实现上，由于I/O设备速度较慢，通常不能与高速的发送速度相匹配，我们一般利用线程进行数据的定时、分时发送。