姓名：章崇文　　学号：202202296　　班级：计算机 222

实验一、嵌入式系统开发环境熟悉

一、实验目的

1. 熟悉嵌入式开发环境，　　掌握实验箱结构及连线方法。

2. 熟悉 Ubuntu 系统进行嵌入式 Linux 开发的基本环境配置方法。

3. 熟悉 arm-linux-gcc 交叉编译器的使用方法。

4. 熟悉 VIM 的使用方法。

二、实验基本要求

1. 熟悉实验箱与主机连接的方法，掌握 Ubuntu 中 IP 地址的配置方法以及挂载共享文件夹的方法。

2. 学会使用 VIM 编写程序。

3. 掌握在虚拟机 Ubuntu 中编译并执行程序的方法。

4. 掌握在虚拟机 Ubuntu 中使用交叉编译器编译程序并在实验箱上运行的方法。

5. 掌握编写 makefile 文件进行编译程序并在对应平台下运行的方法。

三、实验原理

1．GCC 简介

GNU Compiler Collection，通常简称 GCC，是一套由 GNU 开发的编译器集，为什么是编辑器集而不是编译器呢？那是因为它不仅支持 C 语言编译，还支持 C++，Ada，Objective C 等许多语言。另外 GCC 对硬件平台的支持，可以所无所不在，它不仅支持 X86 处理器架构，还支持 ARM, Motorola68000, Motorola8800, AtmelAVR, MIPS 等处理器架构。

2. GCC 的组成结构

GCC 内部结构主要由 Binutils、gcc-core、Glibc 等软件包组成。Binutils：它是一组开发工具，包括连接器，汇编器和其他用于目标文件和档案的工具。关于 Binutils 的介绍可以参考 Binutils 简单介绍。这个软件包依赖于不同的目标机的平台。因为不同目标机的指令集是不一样的，比如 arm 跟 x86 就不一样。

gcc-core：顾明之意是 GCC 的核心部分，这部分是只包含 c 的编译器及公共部分，而对其他语言（C++、Ada 等）的支持包需要另外安装，这也是 GCC 为何如此强大的重

要原因。gcc-core 依赖于 Binutils。

Glibc：包含了主要的 c 库，这个库提供了基本的例程，用于分配内存，搜索目录，读写文件，字符串处理等等。kernel 和 bootloader 不需要这个库的支持。

3. 交叉编译

交叉编译（或交叉建立）是这样一种过程，它在一种机器结构下编译的软件将在另一种完全不同的机器结构下执行。一个常见的例子是在 PC 机上为运行在基于 ARM、PowerPC 或 MIPS 的目标机的编译软件。幸运的是，GCC 使得这一过程所面临的困难要比听起来小得多。GCC 中的一般工具通常都是通过在命令行上调用命令（如 gcc）来执行的。在使用交叉编译的情况下，这些工具将根据它编译的目标而命名。例如，要使用交叉工具链为 ARM

机器编译简单的 HelloWorld 程序，你可以运行如下所示的命令：使用如下命令编译并测试这个代码：arm-linux-gcc –o hello hello.c。

4. arm-linux-gcc

arm-linux-gcc 是基于 ARM 目标机的交叉编译软件。x86 跟 ARM 所使用的指令集是不一样的，所以所需要的 binutils 肯定不一样；上面提到过 gcc-core 是依赖于 binutils 的，自然 ARM 跟 x86 所使用的 gcc-core 包也不一样；glibc 一个 c 库，最终是以库的形式存在于编译器中，自然 ARM 所使用的 glibc 库跟 x86 同样也不一样，其它的依此类推。

四、实验内容

1. 参考《DYV-1 型嵌入式车载多媒体实验箱快速入门手册》，将实验箱与主机连起来，并配置实验箱 IP 地址，然后进行共享文件夹的挂载。

2. 使用 VIM 编写计算 n=1+2+3+…+100 的和。

3. 在虚拟机 Ubuntu 中编译并执行该程序，查看结果。

4. 在虚拟机 Ubuntu 中使用交叉编译器编译该程序，然后在实验箱上运行。

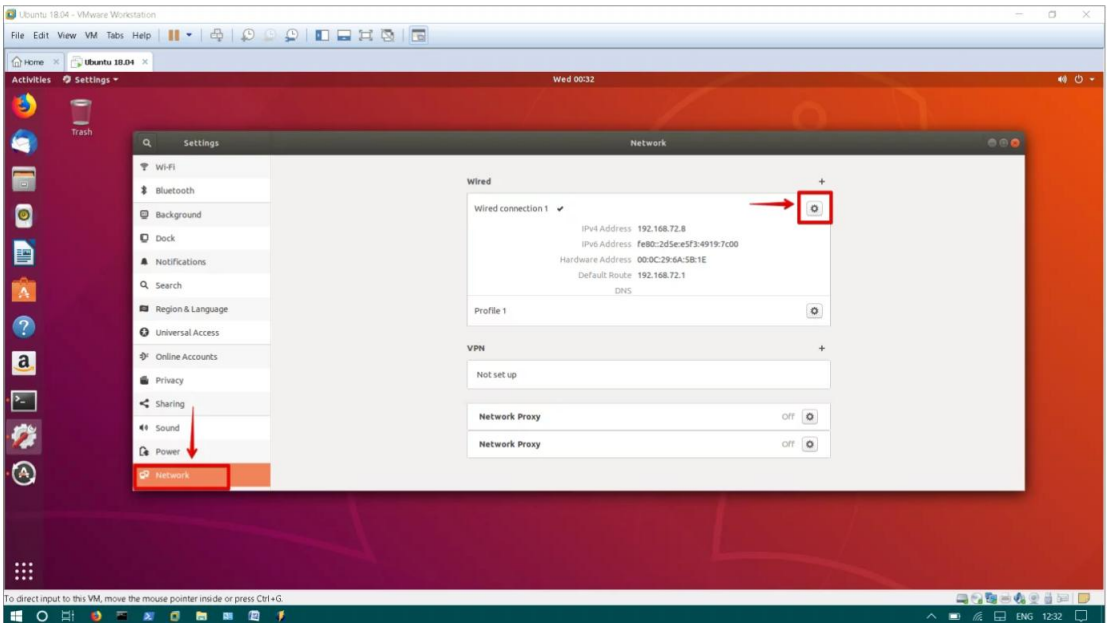
5. 编写 makefile 文件进行编译该程序，然后在对应平台下运行。

五、实验结果及分析：

1. 请叙述几种不同 Ubuntu 的 IP 设置方法。

在 Ubuntu 系统中，配置 IP 地址主要有以下几种方法：

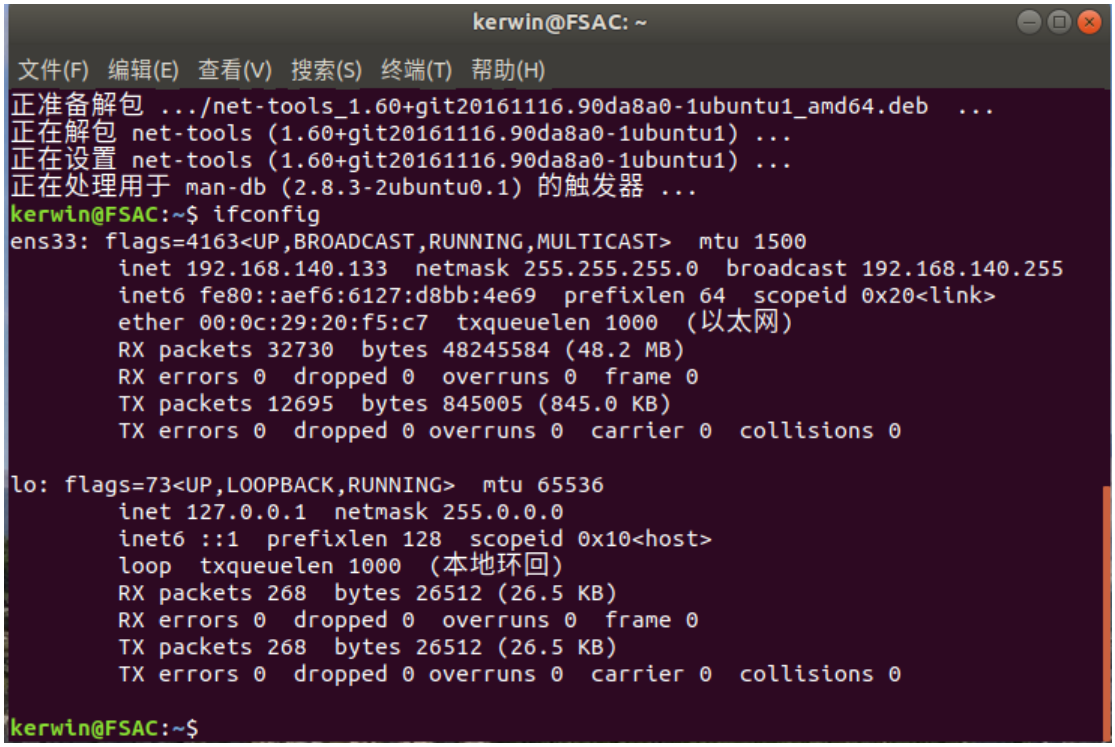
- **图形界面（GUI）配置:**



- 通过桌面环境（如 GNOME, KDE 等）的网络管理器（Network Manager）进行设置。
- 通常在系统设置菜单中找到“网络”或“Network”选项。
- 选择需要配置的有线或无线连接，点击设置（通常是齿轮图标）。
- 在 IPv4（或 IPv6）标签页下，将“方法”或“Method”从“自动（DHCP）”更改为“手动”或“Manual”。
- 输入所需的静态 IP 地址、子网掩码（Netmask）、网关（Gateway）以及可选的 DNS 服务器地址。
- 保存设置并应用，可能需要断开重连网络或重启网络服务。

- **命令行配置（Netplan - 较新版本 Ubuntu）：**
  - Ubuntu 17.10 及之后版本默认使用 Netplan 进行网络配置。
  - 配置文件位于 /etc/netplan/ 目录下，通常是一个 .yaml 文件（如 01-network-manager-all.yaml 或 50-cloud-init.yaml）。
  - 使用文本编辑器（如 sudo vim /etc/netplan/config.yaml）编辑该文件。
  - 修改或添加网络接口（如 eth0, ens33）的配置，设置 dhcp4: no，并添加 addresses: [IP 地址/子网掩码位数]，gateway4: 网关地址，以及 nameservers: { addresses: [DNS1, DNS2] }。
  - 保存文件后，运行 sudo netplan apply 应用配置。

- **命令行配置（ifconfig/ip - 临时):**



  - 可以使用 ip 命令（推荐）或 ifconfig 命令（旧，可能需安装 net-tools）临时配置 IP。
  - 例如：sudo ip addr add 192.168.1.100/24 dev eth0 (添加 IP 和掩码到 eth0 接口)。
  - 例如：sudo ip link set eth0 up (启用接口)。
  - 例如：sudo ip route add default via 192.168.1.1 (添加默认网关)。
  - 这种方法设置的 IP 地址在系统重启后会丢失。

- **命令行配置（/etc/network/interfaces - 较旧版本 Ubuntu 或特定配置):**
  - 在一些较旧版本或未迁移到 Netplan 的系统中，可以通过编辑 /etc/network/interfaces 文件进行静态配置。
  - 例如，为 eth0 配置静态 IP：
  - ```
    auto eth0
    iface eth0 inet static
        address 192.168.1.100
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 8.8.8.8 114.114.114.114
    ```
    ```

- 编辑后，需要重启网络服务，如 sudo systemctl restart networking 或 sudo /etc/init.d/networking restart。

2. 请叙述实验箱系统挂载 Ubuntu NFS 共享文件夹的步骤。

这部分假设已在 Ubuntu 上完成，并且要共享的目录是 /opt/work，并且允许 IP 地址为 10.10.58.89 的实验箱访问)

1. **安装 NFS 服务器软件:** 如果尚未安装，在 Ubuntu 终端运行 sudo apt update && sudo apt install nfs-kernel-server。
2. **创建或指定共享目录:** 确保 /opt/work 目录存在。如果需要创建，则运行 sudo mkdir -p /opt/work。设置合适的权限，例如 sudo chmod 777 /opt/work (实际应用中请根据安全需求设置权限)。
3. **配置 NFS 导出:** 编辑 NFS 配置文件 sudo vim /etc/exports。
4. 添加或修改配置行，指定要共享的目录 (/opt/work) 以及允许访问的客户端（这里是 10.10.58.89）和权限。例如： /opt/work 10.10.58.89(rw,sync,no_subtree_check,no_root_squash)
    - /opt/work: 要共享的 Ubuntu 目录路径。
    - 10.10.58.89: 允许挂载此共享的实验箱 IP 地址（可以替换为网段如 10.10.58.0/24 或用 * 代表所有，但指定 IP 更安全）。
    - (rw,sync,no_subtree_check,no_root_squash): 挂载选项，rw 表示读写，sync 表示同步写入，no_subtree_check 提高效率，no_root_squash 表示客户端 root 用户映射为服务器端 root 用户（需谨慎使用）。
5. **使配置生效:** 运行 sudo exportfs -a。
6. **启动/重启 NFS 服务:** 运行 sudo systemctl restart nfs-kernel-server。
7. **防火墙设置 (如果启用):** 确保 Ubuntu 防火墙允许来自实验箱 IP (10.10.58.89) 对 NFS 端口（通常是 111 和 2049）的访问。

步骤二：在实验箱（客户端）挂载 NFS 共享

1. **确保网络连通:** 确认实验箱能够通过网络访问 Ubuntu 服务器的 IP 地址 10.10.58.89。
2. **安装 NFS 客户端工具 (如果需要):** 确保实验箱系统支持 NFS 挂载。可能需要安装如 nfs-utils 或 nfs-common 的软件包（具体命令依实验箱系统而定）。
3. **创建本地挂载点:** 在实验箱的 /mnt 目录下创建一个用于挂载的空文件夹，根

据你的操作，是 nfs 文件夹。在实验箱终端中运行： mkdir /mnt/nfs

4. **执行挂载命令:** 使用 mount 命令将 Ubuntu 上共享的 /opt/work 目录挂载到实验箱本地的 /mnt/nfs 目录。根据你提供的命令： mount -t nfs -o nolock 10.10.58.89:/opt/work /mnt/nfs
    - -t nfs: 指定文件系统类型为 NFS。
    - -o nolock: 传递挂载选项，nolock 禁用了 NFS 文件锁定，有时用于兼容或解决特定嵌入式环境的问题。
    - 10.10.58.89:/opt/work: 指定 NFS 服务器的 IP 地址 (10.10.58.89) 和服务器上共享的目录路径 (/opt/work)。
    - /mnt/nfs: 指定实验箱本地的挂载点目录。
5. **验证挂载:** 挂载成功后，可以通过以下方式验证：
    - 运行 df -h 或 mount 命令，查看是否有 /mnt/nfs 的挂载信息。
    - 运行 ls /mnt/nfs，应该能看到 Ubuntu 上 /opt/work 目录中的内容。

这里忘记拍挂载成功的照片了，输入 mount -t nfs -o nolock 10.10.58.89:/opt/work /mnt/nfs 没有显示表示挂载成功，后续进入对应文件夹 ls 可以看到相关文件。

3. 请总结 VIM 的工作模式类型及常用命令。

VIM (Vi IMproved) 是一个强大的文本编辑器，主要有以下几种工作模式：

- **工作模式类型:**
    1. **正常模式 (Normal Mode / 命令模式):** 这是 VIM 启动后的默认模式。在此模式下，按键被解释为命令，用于移动光标、删除文本、复制粘贴、进入其他模式等。
    2. **插入模式 (Insert Mode):** 在此模式下，可以像普通编辑器一样输入文本。从正常模式进入插入模式有多种方式（如按 i, a, o 等）。按 Esc 键可以返回正常模式。
    3. **命令行模式 (Command-Line Mode):** 在正常模式下按 :（冒号）进入。可以在底部命令行输入更复杂的命令，如保存文件 (:w)、退出 (:q)、搜索 (:/pattern)、替换 (:s/old/new/g)、设置选项等。按 Enter 执行命令，通常执行后返回正常模式。

4. **可视模式 (Visual Mode):** 用于选择文本块。从正常模式按 v (字符选择)、V (行选择) 或 Ctrl+v (块选择) 进入。选择后可以对选区执行命令（如 d 删除, y 复制）。按 Esc 返回正常模式。

- **常用命令 (主要在正常模式下):**
- 好的，这是格式优化为表格的常用 Vim 命令清单：

类别	命令	说明
模式切换	i	在光标前进入插入模式
	a	在光标后进入插入模式
	o	在当前行下方新建一行并进入插入模式
	O	在当前行上方新建一行并进入插入模式
	<Esc>	从插入模式或可视模式返回正常模式
	:	进入命令行模式
	v, V, <Ctrl>+v	进入可视模式（字符、行、块）
光标移动	h, j, k, l	左、下、上、右
	w	移动到下一个单词开头
	b	移动到上一个单词开头
	0	移动到行首
	$	移动到行尾
	gg	移动到文件第一行
	G	移动到文件最后一行
	nG	移动到第 n 行 (例如: 10G)
	<Ctrl>+f	向下翻页
	<Ctrl>+b	向上翻页

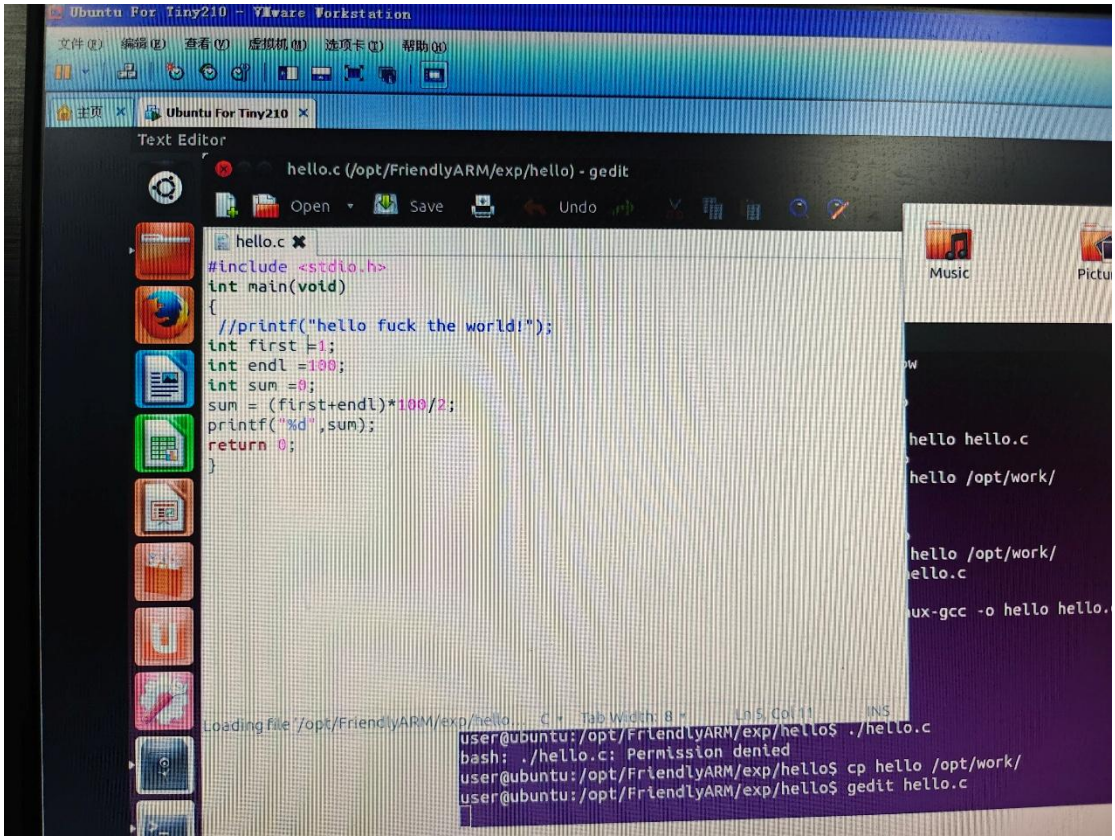
编辑	x	删除光标处的字符
	dw	删除一个单词
	dd	删除当前行
	d<motion>	删除光标移动范围内的文本 (例如: d$ 删除到行尾)
	yw	复制一个单词
	yy	复制当前行 (yank)
	y<motion>	复制光标移动范围内的文本
	p	在光标后粘贴
	P	在光标前粘贴
	u	撤销上一步操作
	<Ctrl>+r	重做撤销的操作
	r<char>	替换光标处的字符为 <char>
	cw	删除一个单词并进入插入模式 (change word)
命令行模式	:w	保存文件
	:q	退出 Vim
	:wq 或 :x	保存并退出
	:q!	强制退出（不保存修改）
	:e <filename>	编辑另一个文件
	:/pattern	向下搜索 pattern
	:?pattern	向上搜索 pattern
	n	跳转到下一个搜索结果

	N	跳转到上一个搜索结果
	:s/old/new/g	在当前行将所有 old 替换为 new
	:%s/old/new/g	在整个文件将所有 old 替换为 new
	:set number	显示行号
	:set nonumber	不显示行号

4. 若想编写一个程序，并且在实验箱里运行。完成该工作，主要分为以下几个步骤?

4.1 使用 Vim 创建源文件进行编程

- 在主机（Ubuntu 虚拟机）上，使用 vim 或其他文本编辑器创建 C 语言源文件 hello.c： vim hello.c
- 在 hello.c 文件中编写计算 0 到 100 累加和的 C 程序代码。

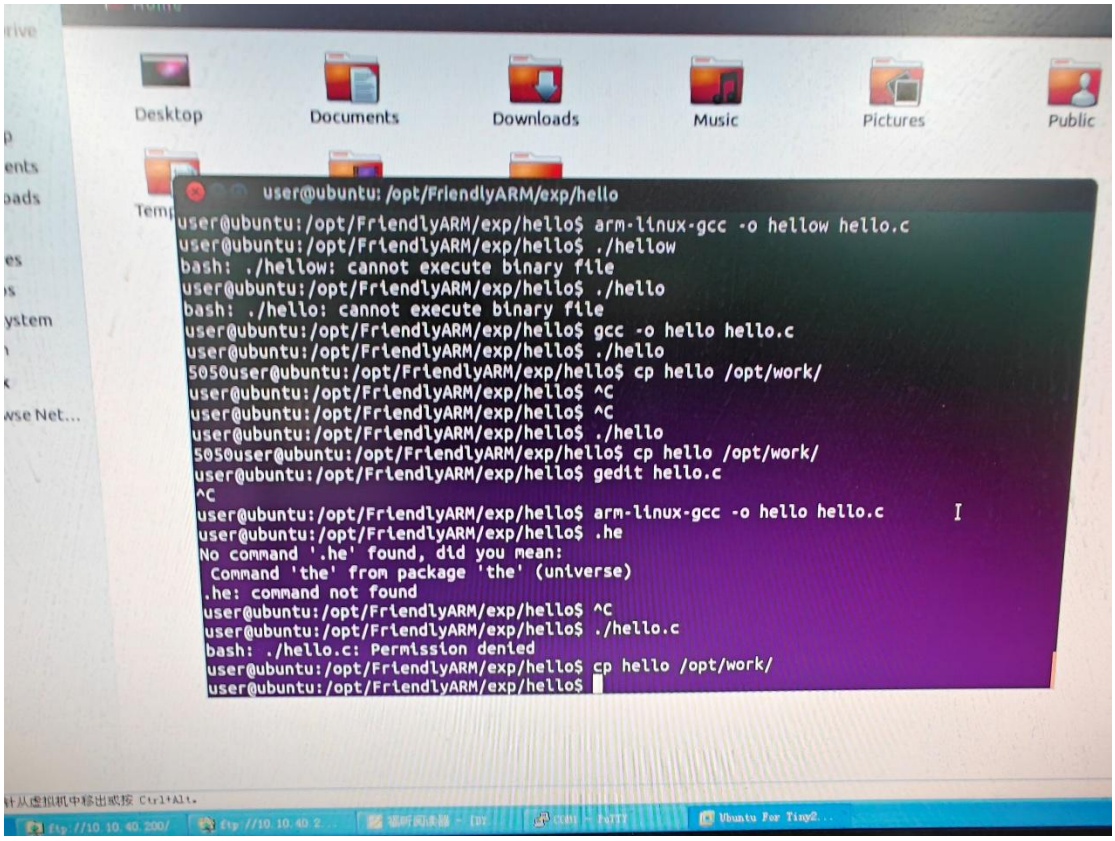


4.2 (可选) 使用本地 GCC 编译测试

- 为了快速检查代码语法和基本逻辑，可以在主机（Ubuntu）上使用本地编译器 gcc 进行编译： gcc hello.c -o hello_host
- 如果编译成功，可以在 Ubuntu 终端运行 ./hello_host 来查看结果是否符合预期 (应该是 5050)。
- **重要提示：** 这一步生成的可执行文件 hello_host 是针对 x86 架构的，**不能**在 ARM 架构的实验箱上运行。此步骤仅用于开发阶段的初步验证。

4.3 使用交叉编译器直接编译

- 在主机（Ubuntu）的终端中，使用 arm-linux-gcc **交叉编译器** 直接编译 hello.c 文件，生成实验箱（ARM 架构）可以运行的可执行文件。 arm-linux-gcc hello.c -o hello_arm
    - arm-linux-gcc: 这是交叉编译器的命令。**注意：** 你实际使用的交叉编译器名称可能带有更具体的前缀，例如 arm-linux-gnueabihf-gcc 等，请使用你环境中正确的命令。
    - hello.c: 你编写的源文件。
    - -o hello_arm: 指定输出的可执行文件名为 hello_arm（你可以使用其他名字，例如 hello，但添加 _arm 或类似后缀有助于区分）。这个 hello_arm 文件是为 ARM 平台编译的。



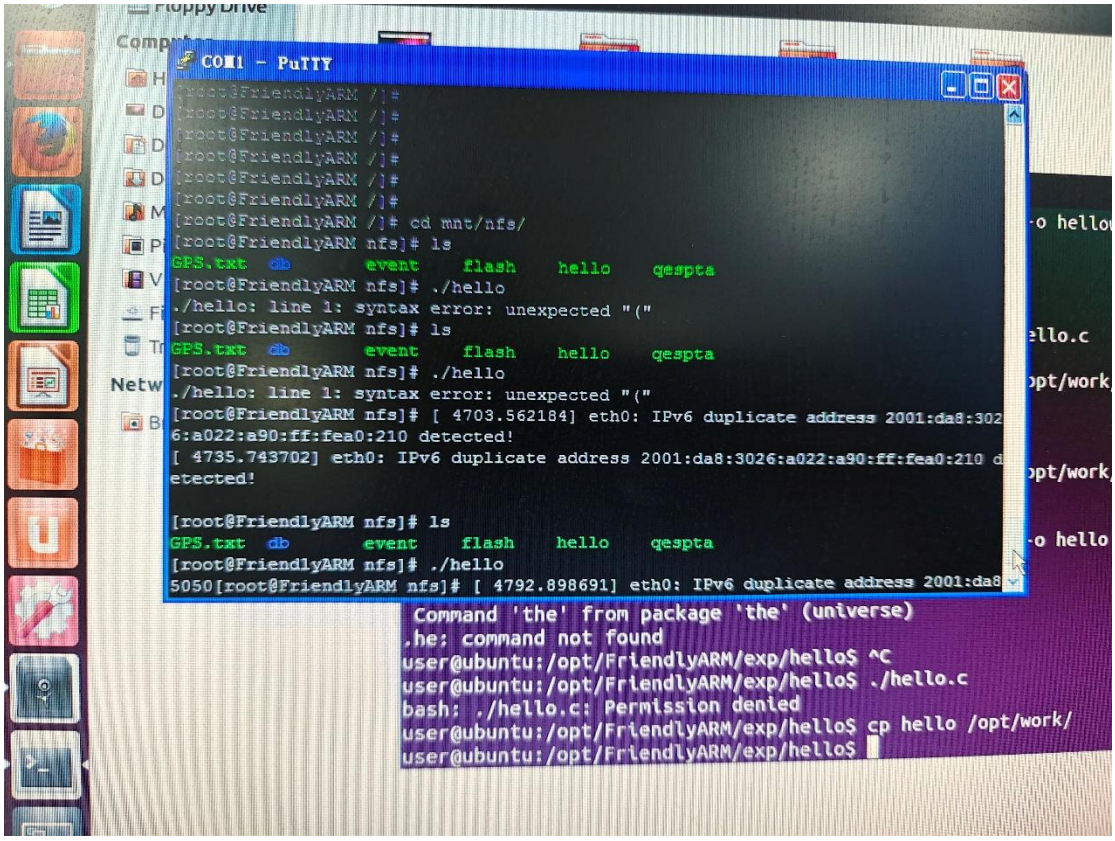
4.4 传输可执行文件到实验箱

- 将上一步交叉编译生成的 **ARM 可执行文件** hello_arm 从主机（Ubuntu）传输到实验箱。
- 可以使用的方法有：
    - **NFS：** 如果配置了 NFS 共享，将 hello_arm 复制到 Ubuntu 的共享目录，然后在实验箱上通过挂载点访问。
    - **SCP (Secure Copy)：** 使用 scp 命令通过网络复制：scp hello_arm 用户名@实验箱 IP 地址:/目标路径/。
    - **TFTP：** 通过 TFTP 服务器和客户端传输。
    - **SD 卡/U 盘：** 将文件复制到存储介质，再插入实验箱。

4.5 在实验箱上运行程序

- 通过串口终端登录到实验箱的操作界面。
- 使用 cd 命令进入存放 hello_arm 文件的目录。
- (通常需要) 赋予文件执行权限： chmod +x hello_arm
- 执行程序： ./hello_arm

- 在实验箱的终端上观察程序的输出，应该会打印出 " 5050"。



这样就完成了在主机上编写代码、直接使用交叉编译器编译、传输到实验箱并最终在实验箱上运行的整个流程。