

**本科生实验报告**

实验课程：操作系统原理实验

任课教师：刘宁

实验题目：编译内核利用已有内核构建OS

专业名称：计算机科学与技术

学生姓名：

学生学号：

实验地点：实验中心B202

实验时间：2025.2.26

**Section 1 实验概述**

本次实验，需要我们熟悉现有Linux内核的编译过程和启动过程，并在自行编译内核的基础上构建简单应用并启动；安装并编译精简的Busybox工具集构建简单的OS，熟悉现代操作系统的构建过程；熟悉编译环境、相关工具集，并能够使用gdb实现内核远程调试。

完成实验之后，可以创建出一个新的虚拟机以及Linux系统，系统拥有基本的开发功能，包括Linux内核和gcc代码编译环境，同时在完成实验之后我们为自己的Linux系统安装好Busybox工具包，该工具包为后续自己搭建操作系统提供着丰富的工具支持。

**Section 2 预备知识与实验环境**

* 预备知识：C/C++语言知识、Linux系统命令行工具
* 实验环境：
  + 虚拟机版本/处理器型号：Virtualbox7.0.6、Ubuntu18.04
  + 代码编辑环境：gedit文本编辑器、gdb调试器
  + 代码编译工具：gcc环境
  + 重要三方库信息：qemu、cmake

**Section 3 实验任务**

* 实验任务1：安装Virtualbox启动虚拟机，完成虚拟机的环境配置，安装Linux环境以及常用的工具库
* 实验任务2：完成Linux内核的编译
* 实验任务3：启动内核并使用gdb调试
* 实验任务4：制作Initramfs并运行，同时再次使用gdb调试
* 实验任务5：安装同时编译并启动Busybox

**Section 4 实验步骤与实验结果**

------------------------- **实验任务1** -------------------------

* 任务要求：安装Virtualbox虚拟机同时安装Ubuntu系统，同时对Linux系统进行初步的配置，具体包括切换下载源，安装编译和编程环境，以及安装后续实验所需要用到的工具.
* 思路分析：

①下载开源免费的Virtualbox软件和Ubuntu的iso文件，在Virtualbox新建 并配置一个新的虚拟计算机界面，

②使用命令行完成Ubuntu系统的安装，同时通过命令行切换成国内的下载 源提升下载速度，

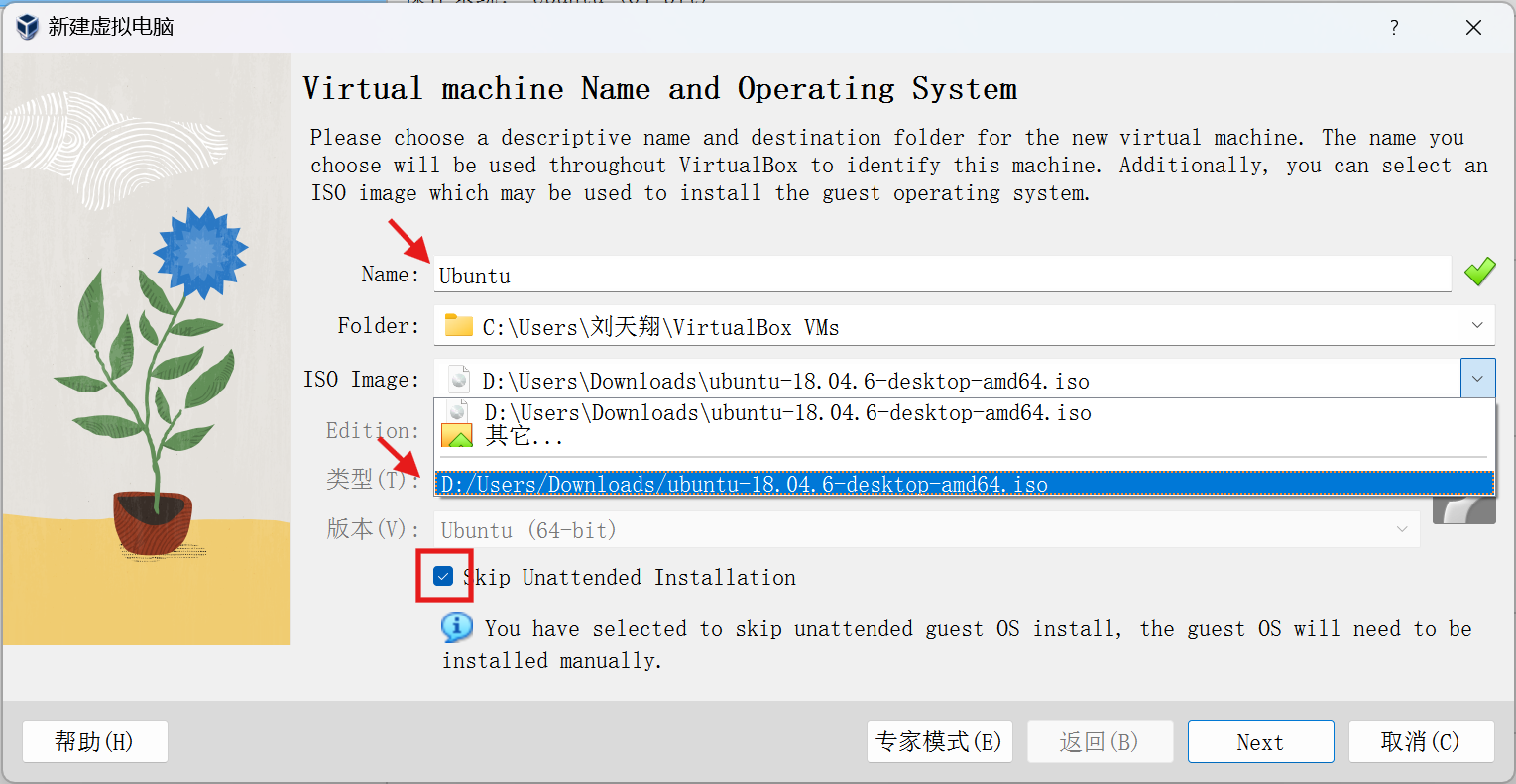
③最后通过sudo apt等相关命令安装后续实验所必须的gcc环境以及后续实 验所需要的常用工具包.

* 实验步骤：

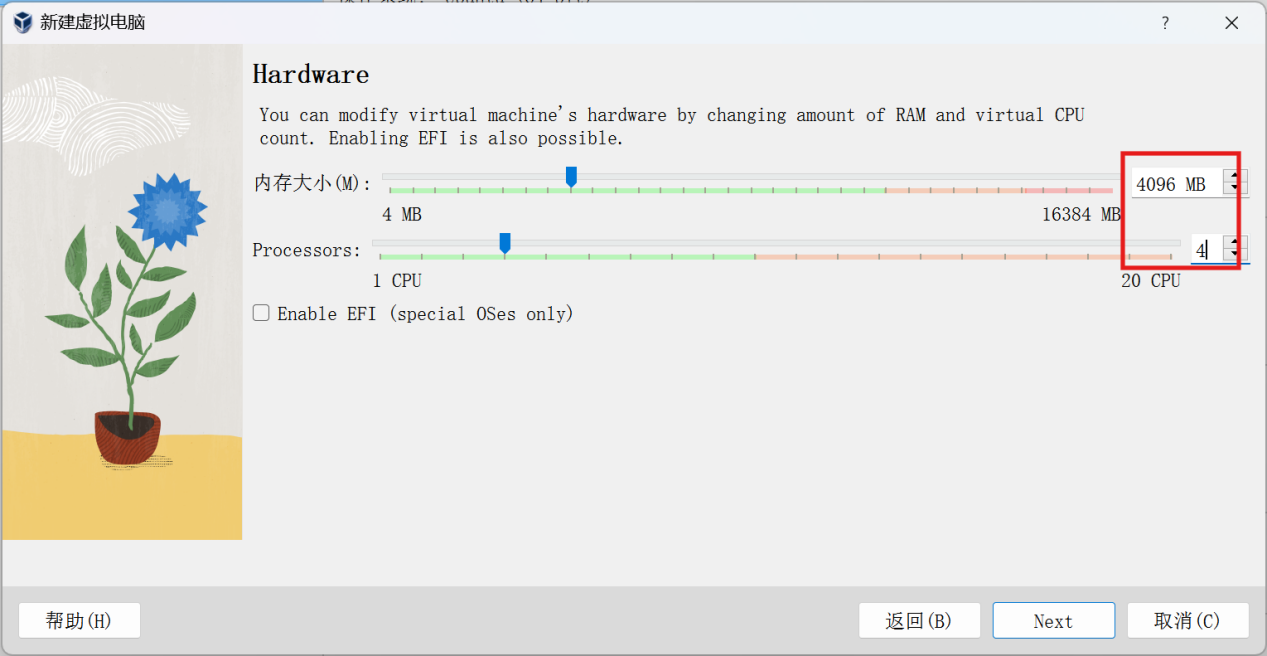
**一、安装Virtualbox（[下载网址](https://download.virtualbox.org/virtualbox/7.0.6/VirtualBox-7.0.6-155176-Win.exe)）以及Ubuntu系统**

安装Virtualbox用于创建虚拟机和安装Linux系统常用版本Ubuntu18.04环境（[下载网址](https://mirrors.tuna.tsinghua.edu.cn/ubuntu-releases/18.04.6/ubuntu-18.04.6-desktop-amd64.iso)），有条件的还可以在自己的安装Linux的物理机方便更好的管理和使用我们的Linux系统。下面是新建和配置一个新的虚拟机的实验过程。

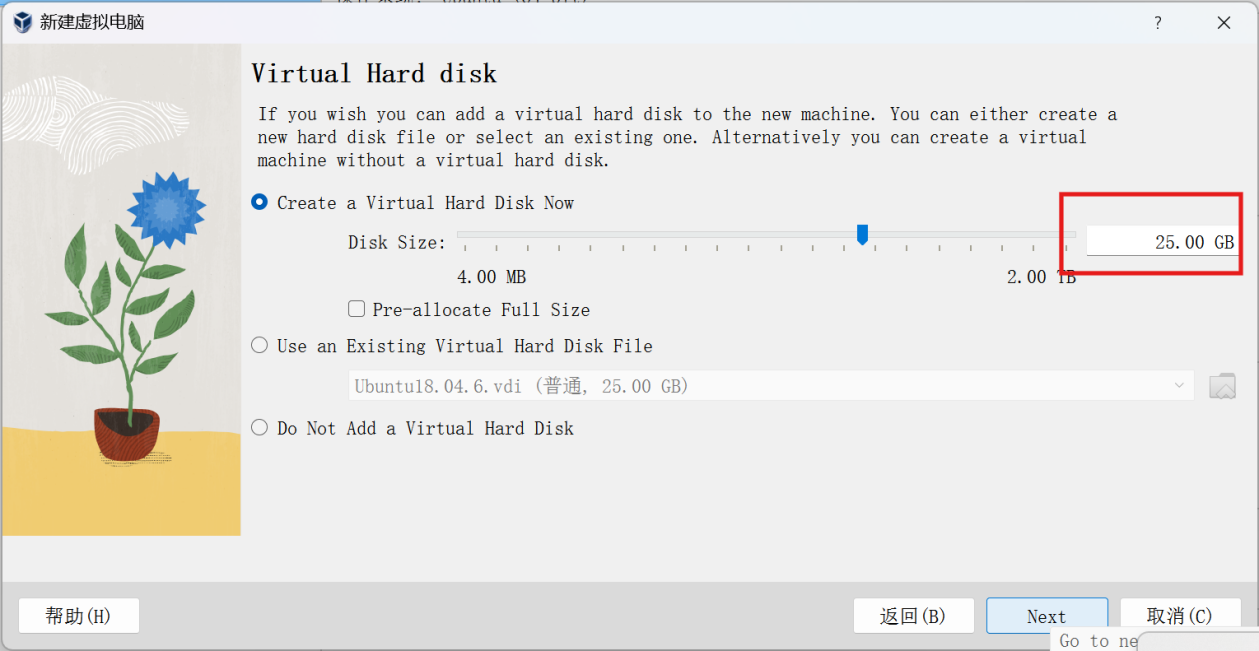
**Step1：**为新建的虚拟机设置一个名字和选择我们之前下载好的Ubuntu18.04的.iso文件.



**Step2：**设置硬件的参数，选择4GB的内存以及4核GPU.



**Step3：**设置虚拟机磁盘的大小，这里选择默认的25GB的外存空间



**Step4：**启动虚拟机之后，我们直接在虚拟机的桌面上找到Ubuntu18.04的程序安装我们的Ubuntu系统，根据安装的提示用自己的名字设置用户名和自己的密码，然后在设置中的Display和Language模块设置成自己想要的分辨率和语言，这样一个虚拟机就配置好了。

**二、更换国内的镜像下载源**

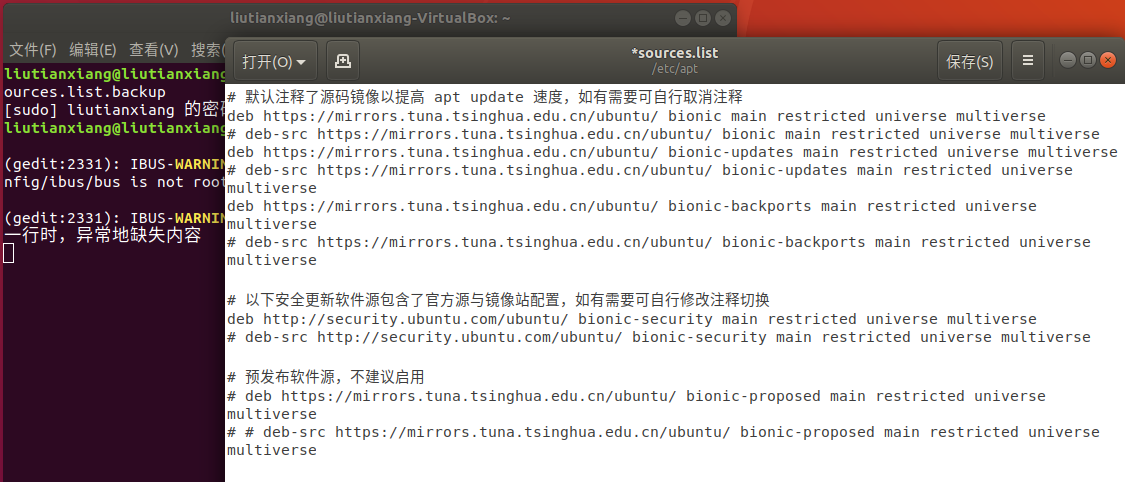
由于Ubuntu的下载源默认是国外的，这影响了我们下载工具包和配置环境的速度，为了提高下载速度，我们需要将下载源更换为国内源。

**Step1：**首先打开Terminal在命令行中输入下面代码备份原来的下载源

sudo mv /etc/apt/sources.list /etc/apt/sources.list.backup

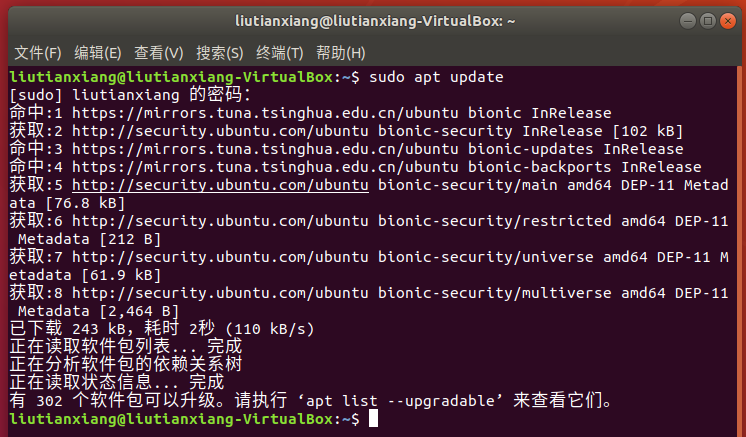
**Step2：**找到[清华的Ubuntu18.04版本的下载源](https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/)，在Terminal中输入如下命令，打开gedit编辑器将下载源复制进/etc/apt/sources.list，然后保存退出。

sudo gedit /etc/apt/sources.list



**Step3：**在命令行中输入如下代码更新apt，检查是否更换为清华的下载源。更新完成之后应该出现如下图所示的字样。

sudo apt update



**三、配置C/C++环境**

为了使我们的虚拟机能够正常编译我们的所写的C/C++程序，需要使用命令行下载gcc的编译环境。

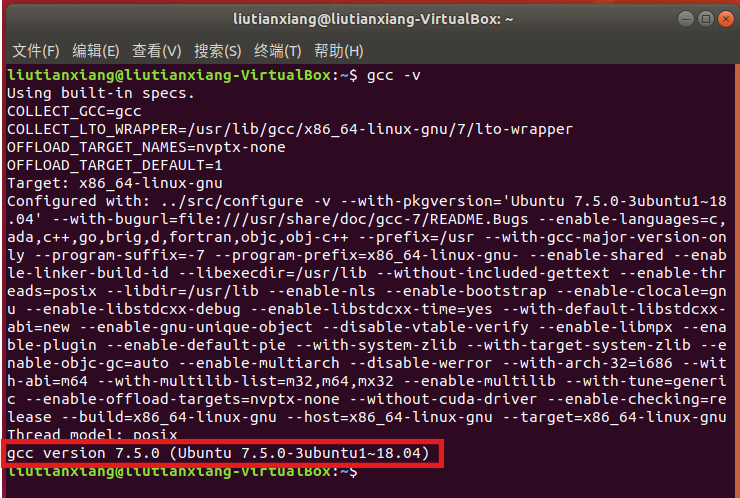
**Step1：**在Terminal中输入如下代码，其中binutils主要用途是汇编和连接，gcc可以编译我们的C/C++程序。

sudo apt install binutils

sudo apt install gcc

**Step2：**安装完之后，输入下面的命令检查gcc是否安装，若输出gcc的版本号则表明安装成功。

gcc -v



**四、安装其他工具**

为了使我们后续的实验能够顺利的进行，我们还需要在我们的虚拟机中安装如下的工具包，用来完成我们接下来的实验。

sudo apt install nasm

sudo apt install qemu

sudo apt install cmake

sudo apt install libncurses5-dev

sudo apt install bison

sudo apt install flex

sudo apt install libssl-dev

sudo apt install libc6-dev-i386

------------------------- **实验任务2** -------------------------

* 任务要求：下载并编译5.10.19版本的Linux内核，
* 思路分析：使用Linux命令行wget下载5.10.19版本的Linux内核，同时通过libncurses5-dev工具配置内核。
* 实验步骤：

**一、下载Linux5.10.x版本的内核**

**Step1：**在虚拟机中创建一个新的文件夹lab1，用来存放我们实验1所需要的文件。mkdir命令用来在当前目录下创建文件夹，cd命令用来打开当前文件夹。

mkdir ~/lab1

cd ~/lab1

**Step2：**下载Linux5.10内核到我们的lab1文件夹，可以到[Linux官网](https://www.kernel.org/)下载，这里也可以使用命令wget来快速的下载

wget [https://mirrors.tuna.tsinghua.edu.cn/kernel/v5.x/](https://mirrors.tuna.tsinghua.edu.cn/kernel/v5.x/linux-5.10.19.tar.xz)

[linux-5.10.19.tar.xz](https://mirrors.tuna.tsinghua.edu.cn/kernel/v5.x/linux-5.10.19.tar.xz)

**Step3：**解压并进入我们下载的内核

xz -d linux-5.10.19.tar.xz

tar -xvf linux-5.10.19.tar

cd linux-5.10.19

**Step4：**配置内核，将内核编译成i386 32位版本。

make i386\_defconfig

make menuconfig

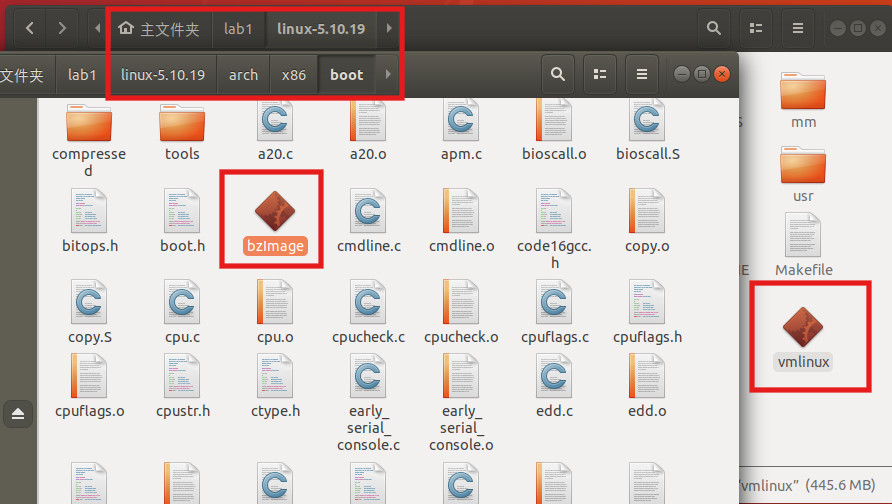
**Step5：**编译内核并检查是否成功

make -j8

* 实验结果展示：

通过执行前述代码，如果编译成功，那么就可以观察到检查已经生成Linux压缩镜像linux-5.10.19/arch/x86/boot/bzImage和符号表linux-5.10.19/vmlinux。

如下图所示结果



------------------------- **实验任务3** -------------------------

* 任务要求：启动Linux内核并对内核进行调试
* 思路分析：通过qeum来启动远程调试同时使用gdb输入指令来对qemu调试窗口进行控制。
* 实验步骤：

**一、启动qemu，使用qemu启动内核并开启远程调试。**

qemu 是一款开源的通用硬件模拟器和虚拟机监控程序，它可以模拟多种不同的硬件平台和操作系统，支持使用者在在计算机上启动并运行操作系统。

**Step1：**使用cd命令，打开在lab1文件夹的目录下，然后输入如下指令打开qemu内核并进行远程调试

cd ~/lab1

qemu-system-i386 -kernel linux-5.10.19/arch/x86/boot/

bzImage -s -S -append "console=ttyS0" -nographic

这时qemu是没有任何输出的，这是因为而qemu在等待gdb输入的指令后才能继续执行，紧接着就打开gdb对qemu进行操控。

**二、启动gdb调试**

为了测试安装的操作系统内核是否能够正常工作，还需要使用基于命令行命令进行调试的gdb程序，使用gdb为start\_kernel函数设置断点，然后使用gdb控制qemu所启动的操作系统内核开始运行。

**Step1：**保持qemu的Terminal，再打开一个新的终端，在命令行中输入如下命令打开lab1文件夹同时启动gdb进行调试

cd ~/lab1

gdb

**Step2：**加载符号表

file linux-5.10.19/vmlinux

**Step3：**连接已经启动的的qemu进行调试

target remote:1234

**Step4：**为start\_kernel函数设置断点

break start\_kernel

**Step5：**输入c运行gdb，控制quem

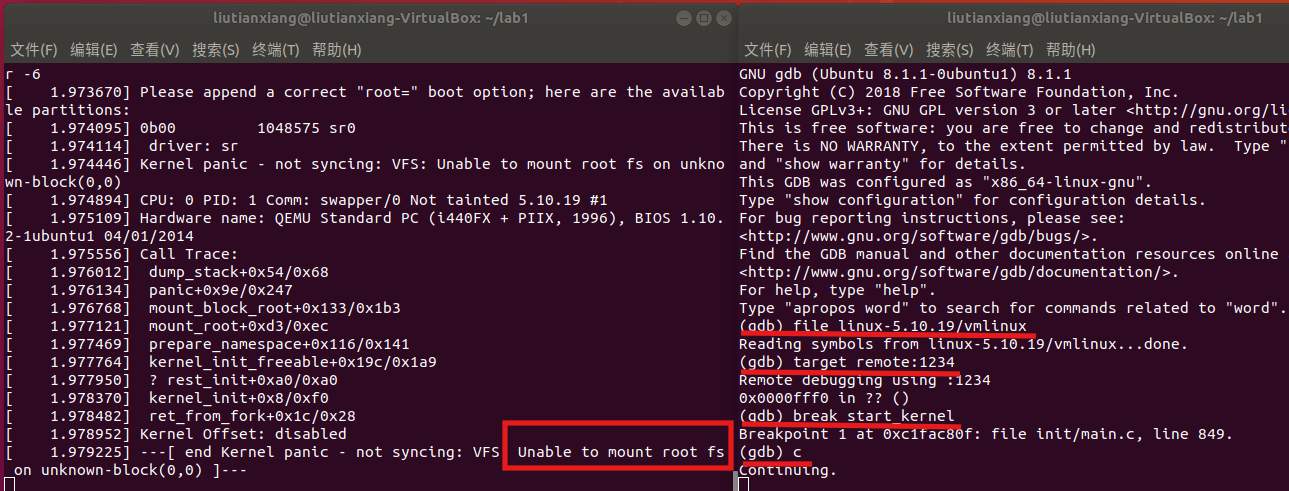
c

* 实验结果展示：通过执行前述代码，可得下图结果。

在继续执行后，最终qemu的输出如下，在qemu虚拟机里运行的Linux系统能成功启动，并且最终以Kernel panic宣告结束。看到call trace打出来的是在initrd\_load的时候出错，观察并比较这次任务打开qemu的命令和实验任务4中启动qemu的命令的区别，发现下面的命令多了一点东西

qemu-system-i386 -kernel linux-5.10.19/arch/x86/boot/bzImage -initrd hwinitramfs -s -S -append "console=ttyS0 rdinit=helloworld" -nographic

也就是说刚刚启动系统的时候只指定了bzImage，并没有指定initrd文件，系统无法mount上initrd (init ram disk) 及其initramfs文件系统，这导致启动时出错。



------------------------- **实验任务4** -------------------------

* 任务要求：为Linux内核制作一个Initramfs
* 思路分析：在上一个任务中发现qemu启动时并没有输出任何东西，这是因为我么启动qemu时没有Initramfs程序，这里通过C语言编写一个简单的Helloworld程序，然后将其编译为可执行文件，使用cpio将其打包，然后作为Initramfs跟着qemu一起启动观察效果，我们就可以看到qemu启动成功并看到我们所编写的程序的输出结果。
* 实验步骤：

**一、编写一个Helloword.c的C语言程序**

**Step1：**打开lab1文件夹使用gedit指令创建并打开helloworld.c文件

cd ~/lab1

gedit helloworld.c

**Step2：**写一个如下的C语言代码，代码可以根据自己的喜好修改。这里选择打印本人的姓名和学号。

#include <stdio.h>

void main()

{

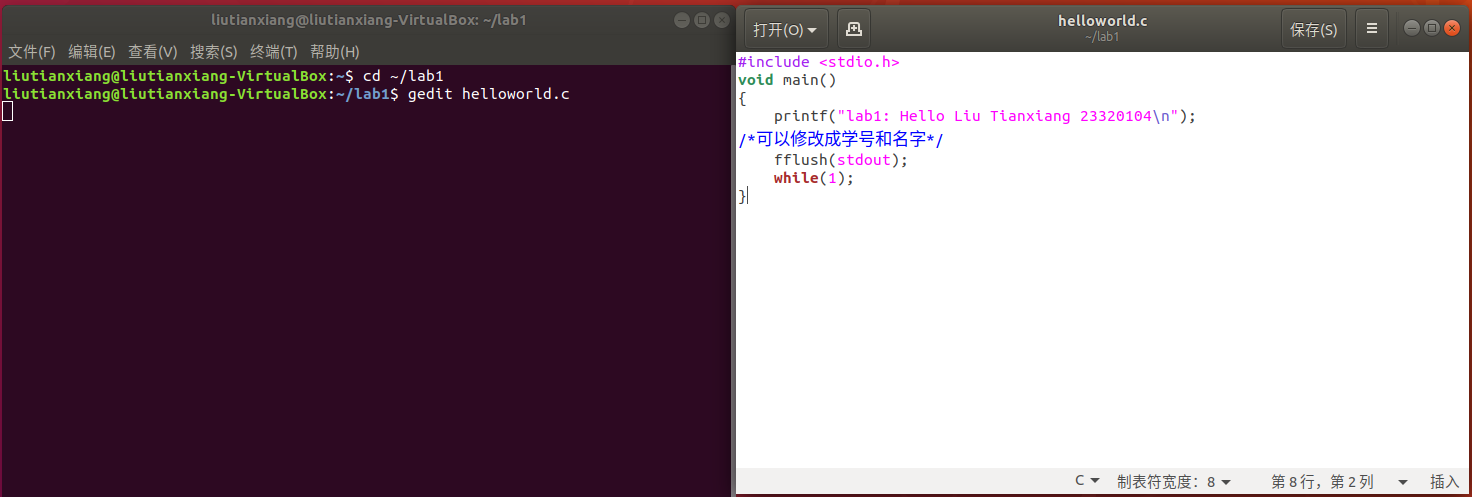
printf("lab1: Hello Liu Tianxiang 23320104\n");

/\*可以修改成学号和名字\*/

fflush(stdout);

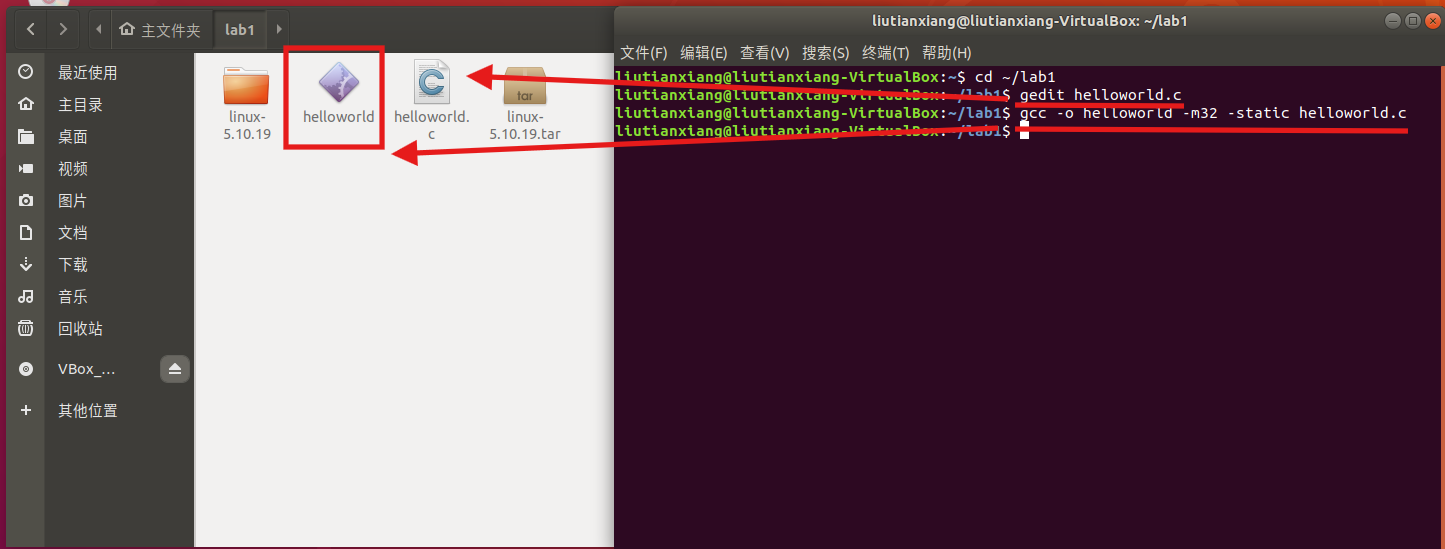
while(1);

}



**Step3：**编译helloworld文件，将他编译成32位可执行文件

gcc -o helloworld -m32 -static helloworld.c



**二、加载Initramfs**

**Step1：**使用cpio打包编译完成的可执行文件

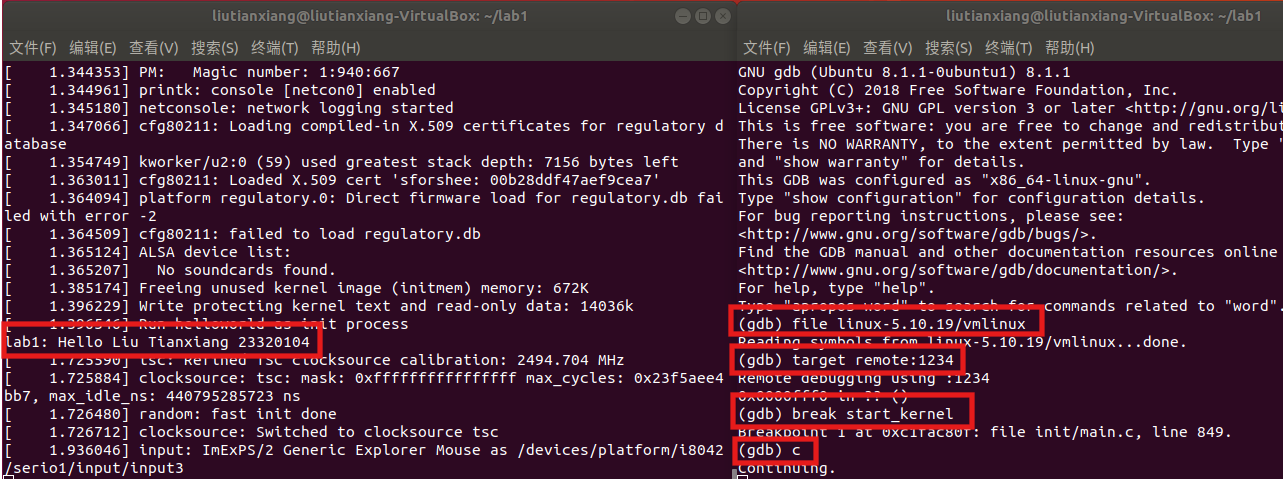
echo helloworld | cpio -o --format=newc > hwinitramfs

**Step2：**带着打包好的文件启动内核并加载Initramfs

qemu-system-i386 -kernel linux-5.10.19/arch/x86/boot/bzImage -initrd hwinitramfs -s -S -append "console=ttyS0 rdinit=helloworld" -nographic

**Step3：**重复实验任务3的gdb的调试过程，观察qemu的输出。

* 实验结果展示：通过执行前述代码，可得下图结果，可以看到qemu输出了我们所编写的程序的结果。



------------------------- **实验任务5** -------------------------

* 任务要求：编译并启动Busybox工具包
* 思路分析：

①通过命令行指令下载busybox工具包，

②同时通过libncurses5-dev工具包对busybox进行编译，

③再通过实验4的步骤为其制作一个Initramfs，

④最后加载busybox，我们可以使用ls来查看当前文件夹的情况。

* 实验步骤：

**一、下载并解压Busybox**

**Step1：**下载Busybox，首先需要去官网或者其他方法将busybox的安装包下载到我们的虚拟机系统中，这里在命令行使用了wget命令来下载busybox安装包到我们的lab1文件夹中。

cd ~/lab1

wget <https://busybox.net/downloads/busybox-1.33.0.tar.bz2>

**Step2：**使用tar命令解压busybox安装包

tar -xf Busybox\_1\_33\_0.tar.gz

**二、编译Busybox**

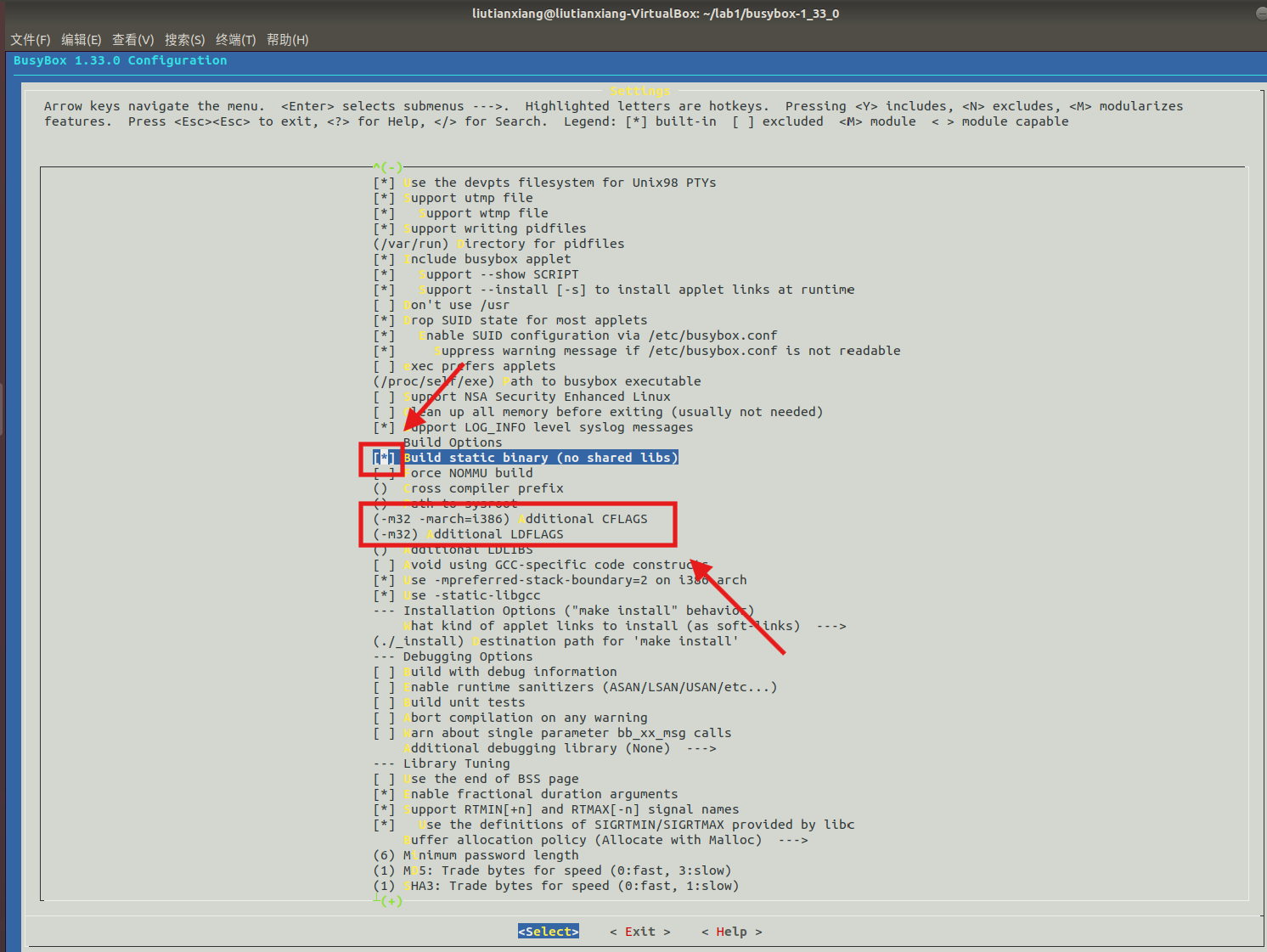
**Step1：**首先我们需要配置busybox，进入到解压出的busybox文件夹目录下，输入如下的指令对busybox进行配置。

cd busybox-1\_33\_0

make defconfig

make menuconfig

进入settings后，在Build static binary(no shared libs)处输入Y勾选，然后ENTER，选中() Additional CFLAGS并键入-m32 -march=i386，以及在() Additional LDFLAGS 键入(-m32)，如下图所示：



**Step2：**输入如下命令编译busybox

make -j8

make install

**三、为Busybox制作一个Initramfs**

**Step1：**使用mkdir命令在lab1中新建一个mybusybox 的文件夹，然后输入如下的命令行命令，将安装在\_install目录下的文件和目录取出放在~/lab1/mybusybox处。

cd ~/lab1

mkdir mybusybox

mkdir -pv mybusybox/{bin,sbin,etc,proc,sys,usr/{bin,sbin}}

cp -av busybox-1\_33\_0/\_install/\* mybusybox/

cd mybusybox

**Step2：**为Initramfs创建一个init程序，使用gedit命令创建一个init的新建文本，

gedit init

然后将下面的Shell脚本代码复制进去后保存并退出。

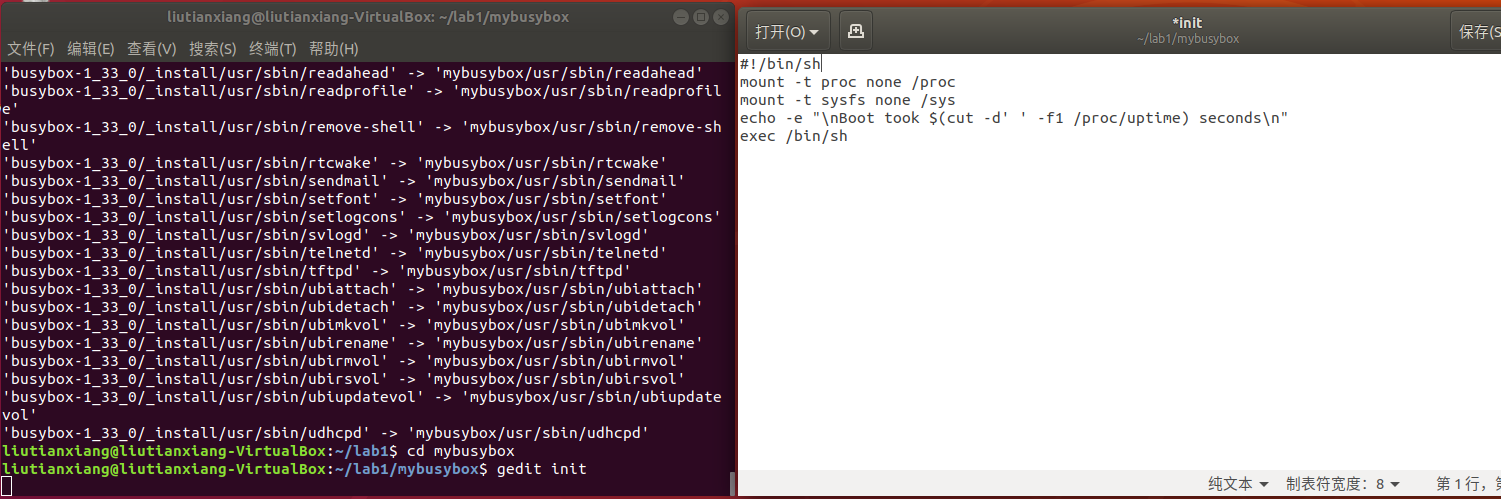
#!/bin/sh

mount -t proc none /proc

mount -t sysfs none /sys

echo -e "\nBoot took $(cut -d' ' -f1 /proc/uptime) seconds\n"

exec /bin/sh



**Step3：**在命令行输入下面的命令，加上执行权限，使用chmod修改init文件权限，增加user使用这个可执行文件的权限。

chmod u+x init

**Step4：**将mybusybox中的内容打包归档成cpio文件，以供Linux内核做Initramfs启动执行。

find . -print0 | cpio --null -ov --format=newc | gzip -9 > /lab1/initramfs-busybox-x86.cpio.gz

**四、加载Busybox**

**Step1：**在命令行输入下面的命令，带着前面打包好的cpio文件启动qemu

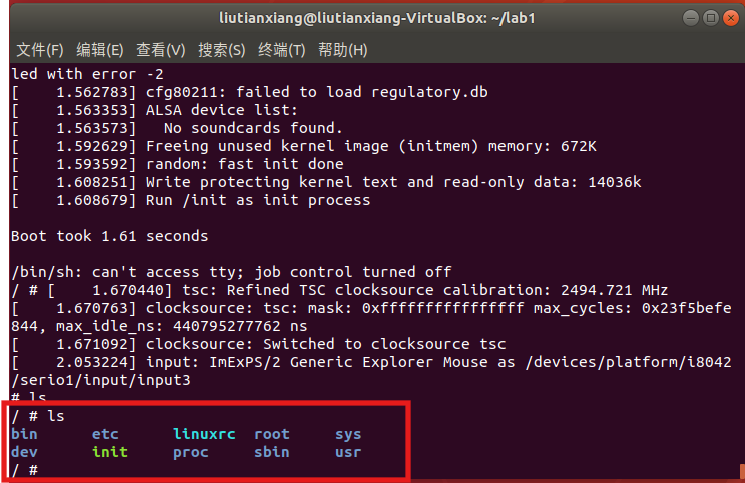
cd ~/lab1

qemu-system-i386 -kernel linux-5.10.19/arch/x86/boot/bzImage -initrd initramfs-busybox-x86.cpio.gz -nographic -append "console=ttyS0"

**Step2：**输入ls命令查看当前文件夹。

ls

* 实验结果展示：通过执行前述代码，可得下图结果，



**Section 5 实验总结与心得体会**

本在实验中由于第一次接触虚拟机和Linux系统，对于虚拟机的创建和配置方法不是非常熟悉，为了能够正确的配置新建的虚拟机，在查阅资料得知了创建虚拟机的常见配置和原理，成功的创建了自己的第一个Linux虚拟机。

通过这次实验学习到了很多以前没有接触过的知识，通过实验了解到了关于操作系统的一些初步的概念，比如了解到了操作系统的内核态和用户态、操作系统的功能等等。

同时了解到了Initramfs的概念，Initramfs在系统启动初期过程中被加载到内存中，执行一些驱动程序或者为操作系统内核提供一些必要的文件。

以及了解到了qemu配合gdb的调试方式，qemu可以模拟多种不同的硬件平台和操作系统，帮助我们运行操作系统，实际上qemu可以支持在同一台物理机上操作不同的操作系统。而gdb不同于之前遇到的IDE图形化debug方式，他提供了一种基于命令行的调试方式，更加的简洁和方便。

最后通过查阅资料，了解到了Busybox的用处，Busybox工具包，为我们后续自己搭建一个简单的操作系统提供丰富的工具，利用Busybox多种功能，可以一步一步的搭建一个简单基础的操作系统。

这次实验不仅提高了我对操作系统概念的知识的理解，同时也学习到了很多有用的技巧，比如通过本次实验，了解到了有关于虚拟机和Linux系统的很多知识，学会了创建Linux虚拟机的方法以及Linux终端命令行的一些常用的命令和技巧，比如sudo apt、cd、wget等等常用的命令行的命令，体会到了相对于图形用户界面操作（GUI）命令行控制在操控计算机的强大和便利之处。

**Section 6 附录：参考资料清单**

1. [技术|完整指南：使用 VirtualBox 在 Windows 上安装 Ubuntu](https://linux.cn/article-15472-1.html)
2. <https://wangchujiang.com/linux-command/hot.html>

**Section 7 附录：代码清单**

【实验任务4】完整的hello world.c程序如下：

#include <stdio.h>

void main()

{

printf("lab1: Hello Liu Tianxiang 23320104\n");

/\*可以修改成学号和名字\*/

fflush(stdout);

while(1);

}

【实验任务5】完整的init程序如下：

#!/bin/sh

mount -t proc none /proc

mount -t sysfs none /sys

echo -e "\nBoot took $(cut -d' ' -f1 /proc/uptime) seconds\n"

exec /bin/sh