

本科生实验报告

姓名：陈雪玮

学号：21307387

1.实验要求

- 1. 独立完成实验5个部份环境配置、编译Linux内核、Qemu启动内核并开启远程调试、制作Initramfs和编译并启动Busybox。
- 2. 编写实验报告、结合实验过程来谈谈你完成实验的思路和结果，最后需要提供实验的5个部份的程序运行截屏来证明你完成了实验。

2.实验步骤/实验过程/实验结果

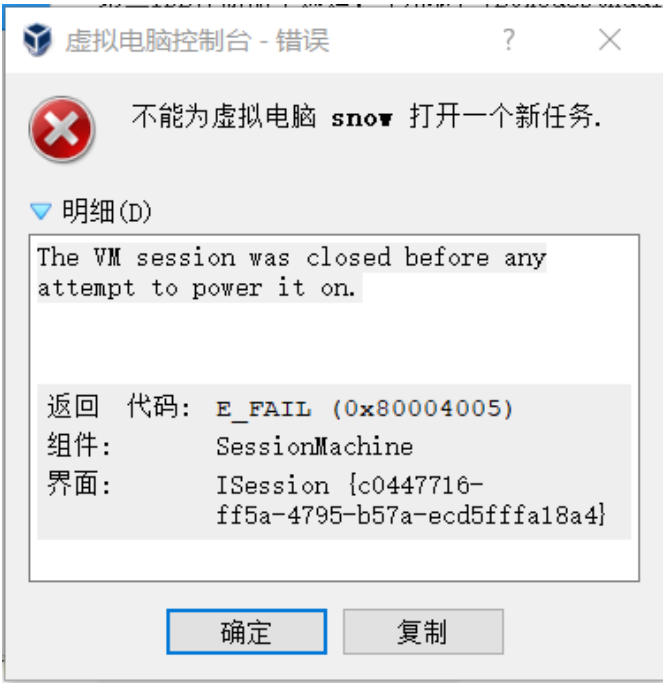
安装虚拟机并启动

这一步是启动虚拟机。

使用VirtualBox作为虚拟机的启动环境，操作系统为Ubuntu18.04。名称为snow，操作系统的用户名也为snow。



启动过程中出现了“不能为虚拟电脑打开一个新任务”的问题，解决方法是设置为每次都以管理员身份运行。



启动虚拟机后按照步骤安装即可。

环境配置

这一步是配置好下载环境和代码编译环境，以及为后面的步骤准备好工具集。

用到的Linux命令

sudo：以系统管理者的身份执行指令，也就是说，经由 sudo 所执行的指令就好像是 root 亲自执行。

mv：（move file）用来为文件或目录改名、或将文件或目录移入其它位置。

apt：（Advanced Packaging Tool）是一个在 Debian 和 Ubuntu 中的 Shell 前端软件包管理器。提供了查找、安装、升级、删除某一个、一组甚至全部软件包的命令，而且命令简洁而又好记。命令执行需要超级管理员权限(root)，与sudo一起使用。

gedit：启动GNOME桌面环境下的文本编辑器gedit。

mkdir：（make directory）用于创建目录。

cd：（change directory）用于切换当前工作目录。另外，~ 也表示为 home 目录 的意思，. 则表示目前所在的目录，.. 则表示目前目录位置的上一层目录。

tar：（tape archive ）命令用于备份文件。用来建立，还原备份文件的工具程序，它可以加入，解开备份文件内的文件。

cpio：cpio 是用来建立，还原备份档的工具程序，它可以加入，解开 cpio 或 tar 备份档内的文件。

换源

为了提高下载速度，将Ubuntu默认的国外下载源更换为国内的下载源。这里使用中国科技大学的镜像源。

Ubuntu (/etc/apt/sources.list):

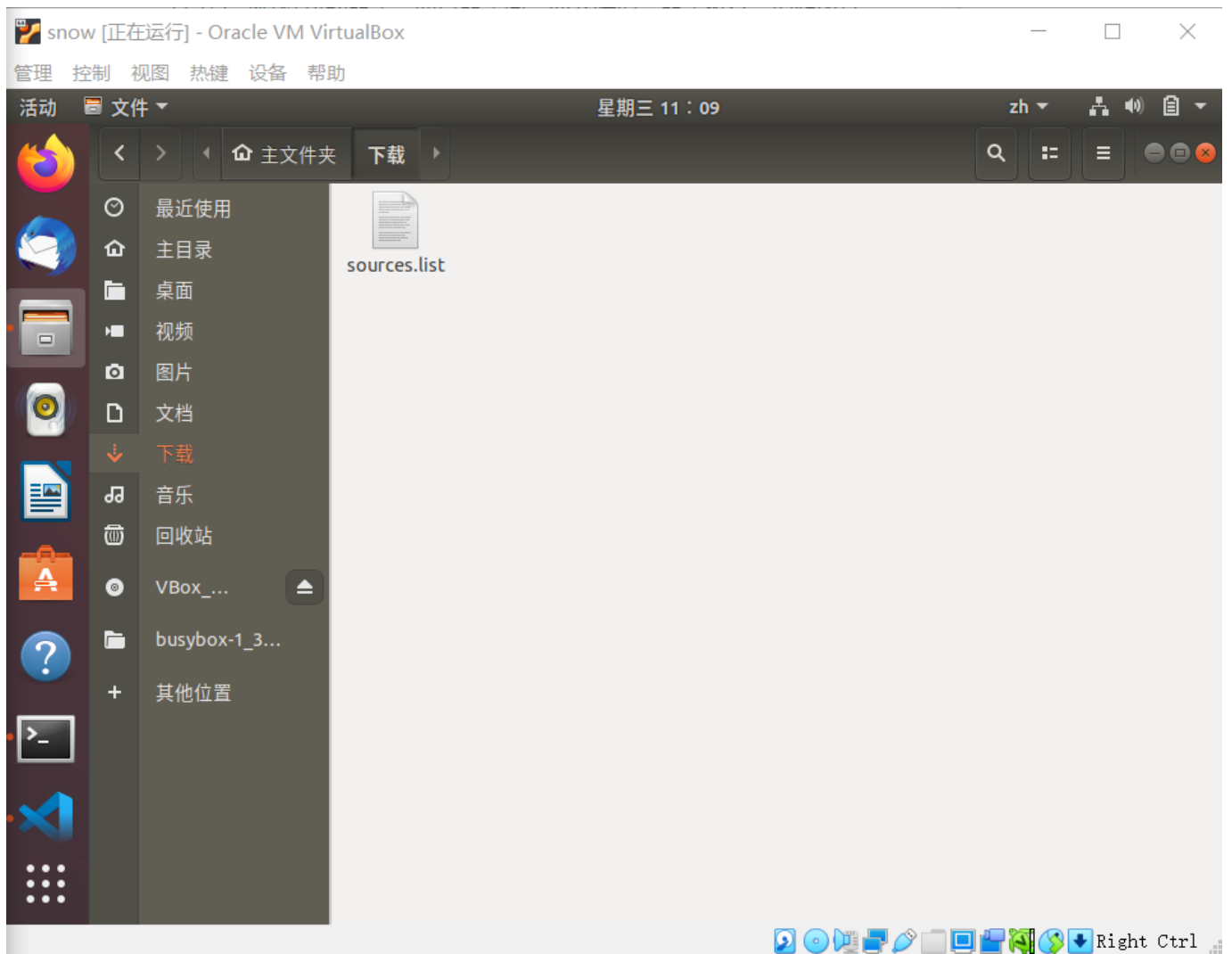
```
deb https://mirrors.ustc.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic main restricted universe multiverse

deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-security main restricted universe multiverse

deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse

deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse

## Not recommended
# deb https://mirrors.ustc.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
# deb-src https://mirrors.ustc.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
```



换源成功

```
snow@snow-VirtualBox: ~/下载
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
snow@snow-VirtualBox:~/下载$ sudo cp sources.list /etc/apt
[sudo] snow 的密码:
snow@snow-VirtualBox:~/下载$ sudo apt-get update
命中:1 https://mirrors.ustc.edu.cn/ubuntu bionic InRelease
命中:2 https://mirrors.ustc.edu.cn/ubuntu bionic-security InRelease
命中:3 https://mirrors.ustc.edu.cn/ubuntu bionic-updates InRelease
命中:4 https://mirrors.ustc.edu.cn/ubuntu bionic-backports InRelease
hythmbox: http://packages.microsoft.com/repos/code stable InRelease [3,023 B]
已下载 3,023 B，耗时 1秒 (2,547 B/s)
正在读取软件包列表... 完成
snow@snow-VirtualBox:~/下载$
```

配置C/C++环境

这一步骤是为了配置C/C++环境。先搞清楚一些名词是什么，以及为什么要安装这些工具。

GUN：GNU的全称是“GNU’s Not Unix!”，是一个自由的操作系统，其内容软件完全以GPL方式发布。

GPL： 全称是 GNU General Public License ，被翻译为 **GNU通用公共许可协议**。

GCC： 全称是 GNU Compiler Collection，是GUN计划制作的一种优化编译器。

binutils： GNU Binutils，是 GNU Binary Utilities 的简写，一般简称为 Binutils。 **中文可以翻译为GNU 的二进制工具集。**

binutils安装成功

```
snow@snow-VirtualBox:~$ sudo apt install binutils
[sudo] snow 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
binutils 已经是最新版 (2.30-21ubuntu1~18.04.8)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
。
```

gcc安装成功

```
snow@snow-VirtualBox:~$ sudo apt install gcc
[sudo] snow 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
gcc 已经是最新版 (4:7.4.0-1ubuntu2.3)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
```

```
snow@snow-VirtualBox: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

snow@snow-VirtualBox:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1~18.04' --with-bugurl=file:///usr/share/doc/gcc-7/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++ --prefix=/usr --with-gcc-major-version-only --program-suffix=-7 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-bootstrap --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-libmpx --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib --enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
snow@snow-VirtualBox:~$
```

安装其他工具

同样，我们先搞清楚这些工具是什么。

qemu：以GPL许可证分发源码的虚拟操作系统模拟器。开虚拟机用的。

cmake：跨平台的编译工具，能够输出各种各样的makefile或者project文件。

libncurses5-dev：图形函数库，作用是可以在终端内绘制简单的图形用户界面。

bison：属于 GNU 项目的一个语法分析器生成器。

flex：一个生成词法扫描器的工具。

libssl-dev: 是OpenSSL通用库, OpenSSL是广泛使用的商业级SSL工具, SSL也即Secure Socket Layer, 是由网景公司为了传输敏感数据而提出的协议。SSL使用私钥加密传输的数据, 防止被窃听。

libc6-dev-i386: 用于 i386 架构内核/系统的 libc 的 i386 版本。这是用于运行为 i386 系统构建的包。

gcc-multilib : 主要是用于**支持交叉编译** (cross compiling) , 即编译出来的程序是用来在其他处理器平台上运行的。

g++-multilib: 交叉编译器。

nasm: NASM是一个汇编器的名称, 全称是Netwide Assembler, 支持x86与x64架构的CPU(注意不支持ARM架构)。

我们需要利用qemu启动编译好的i386内核, 并且在上面创建我们的操作系统, 而程序的编译过程是预处理、编译、汇编、链接, 因此我们需要以上工具实现内核搭建和程序编译。

这些工具安装成功

```
sn0w@sn0w-VirtualBox:~$ sudo apt install qemu
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
qemu 已经是最新版 (1:2.11+dfsg-1ubuntu7.41)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
。
sn0w@sn0w-VirtualBox:~$ sudo apt install cmake
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
cmake 已经是最新版 (3.10.2-1ubuntu2.18.04.2)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
。
sn0w@sn0w-VirtualBox:~$ sudo apt install libncurses5-dev
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
libncurses5-dev 已经是最新版 (6.1-1ubuntu1.18.04)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
。
sn0w@sn0w-VirtualBox:~$ sudo apt install bison
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
bison 已经是最新版 (2:3.0.4.dfsg-1build1)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
。
sn0w@sn0w-VirtualBox:~$ sudo apt install flex
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
flex 已经是最新版 (2.6.4-6)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
```



```
snow@snow-VirtualBox:~$ sudo apt install libssl-dev
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
libssl-dev 已经是最新版 (1.1.1-1ubuntu2.1~18.04.21)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
。

snow@snow-VirtualBox:~$ sudo apt install libc6-dev-i386
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
libc6-dev-i386 已经是最新版 (2.27-3ubuntu1.6)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
。

snow@snow-VirtualBox:~$ sudo apt install gcc-multilib
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
gcc-multilib 已经是最新版 (4:7.4.0-1ubuntu2.3)。
gcc-multilib 已设置为手动安装。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 0 个软件包未被升级
。
```

```
snow@snow-VirtualBox:~$ sudo apt install g++-multilib
[sudo] snow 的密码：
对不起，请重试。
[sudo] snow 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件：
  g++ g++-7 g++-7-multilib lib32stdc++-7-dev libstdc++-7-dev
  libx32stdc++-7-dev
建议安装：
  gcc-7-doc libstdc++6-7-dbg lib32stdc++6-7-dbg libx32stdc++6-7-dbg
  libstdc++-7-doc
下列【新】软件包将被安装：
  g++ g++-7 g++-7-multilib g++-multilib lib32stdc++-7-dev libstdc++-7-dev
  libx32stdc++-7-dev
升级了 0 个软件包，新安装了 7 个软件包，要卸载 0 个软件包，有 274 个软件包未被升
级。
需要下载 12.4 MB 的归档。
解压缩后会消耗 60.0 MB 的额外空间。
您希望继续执行吗？ [Y/n] Y
获取:1 https://mirrors.ustc.edu.cn/ubuntu bionic-security/main amd64 libstdc++-7
-dev amd64 7.5.0-3ubuntu1~18.04 [1,471 kB]
获取:2 https://mirrors.ustc.edu.cn/ubuntu bionic-security/main amd64 g++-7 amd64
7.5.0-3ubuntu1~18.04 [9,697 kB]
```

vscode安装成功



nasm-2.15安装成功

可以使用命令wget从网络上下载安装得到

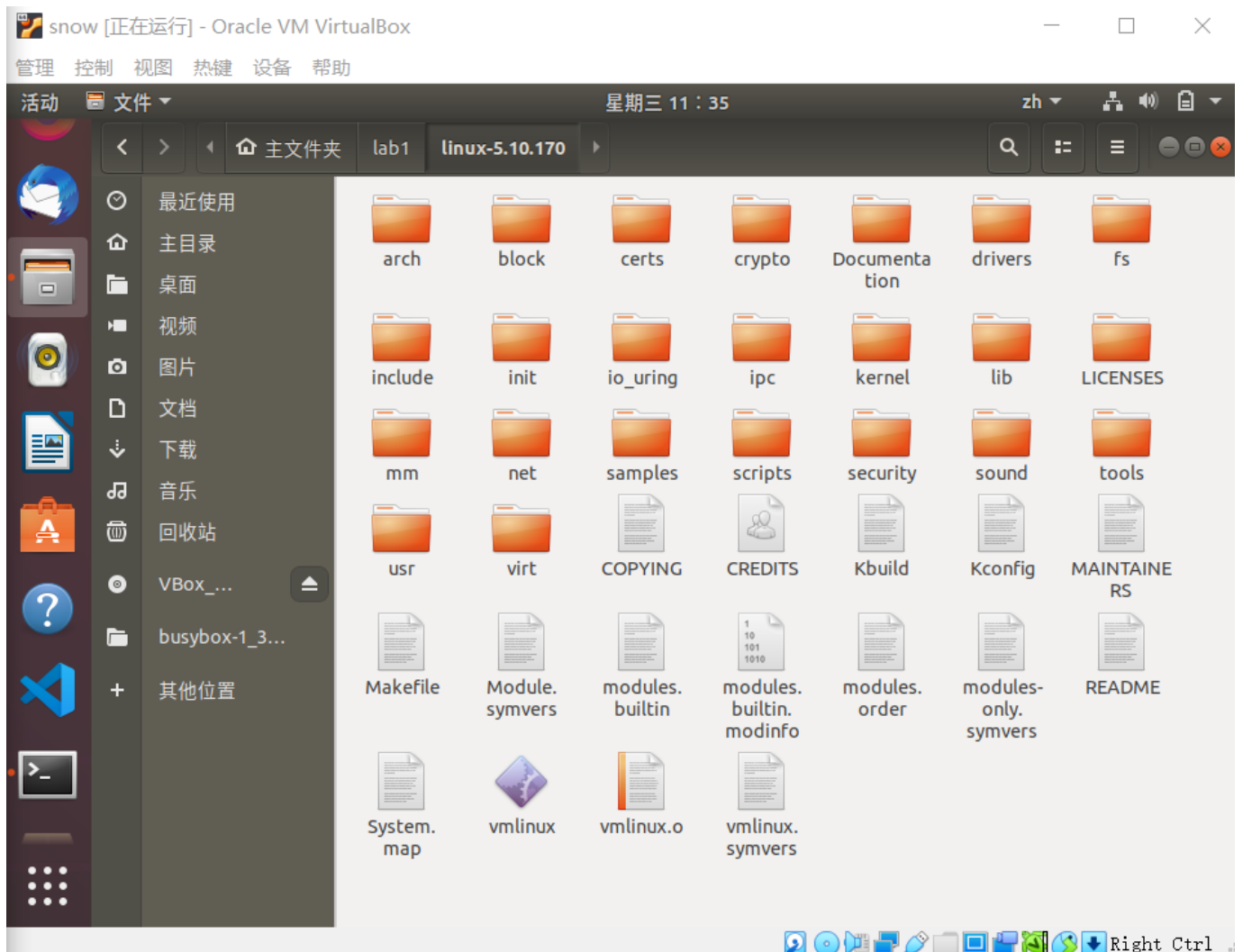
```
snow@snow-VirtualBox:~$ nasm -v  
NASM version 2.15 compiled on Mar  4 2023
```

编译Linux内核

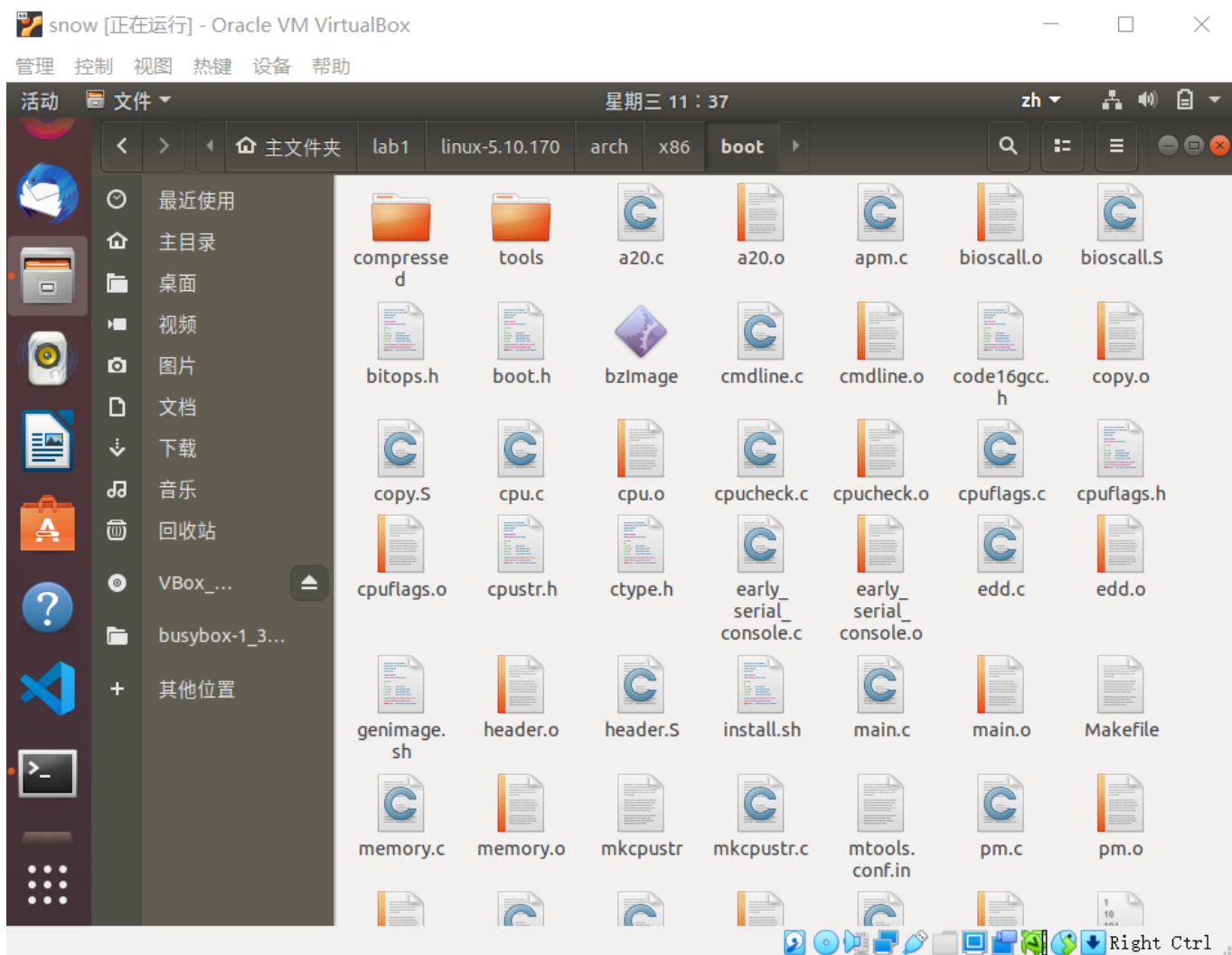
这一步是将下载来的Linux内核编译成i386 32位版本。

编译成功如下：

符号表linux-5.10.19/vmlinux



压缩镜像linux-5.10.19/arch/x86/boot/bzImage



启动内核并调试

这一步是用qemu启动编译好的i386内核，并通过gdb对其调试。

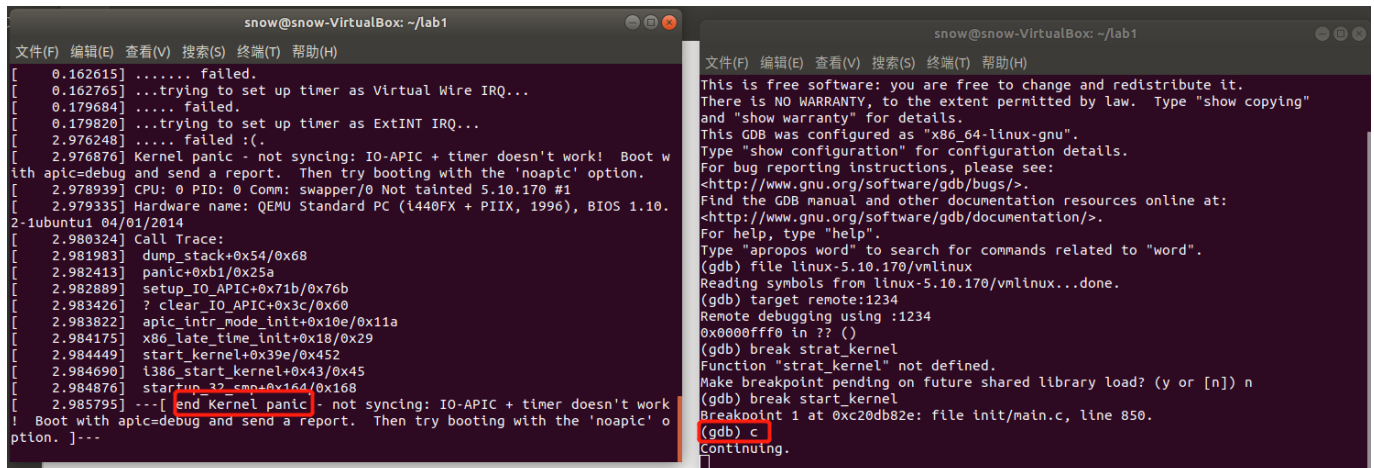
首先了解要用到的gdb是什么

GDB 全称“GNU symbolic debugger”，一般来说，GDB主要帮助我们完成以下四个方面的功能：

1. 启动你的程序，可以按照你的自定义的要求随心所欲的运行程序。
2. 在某个指定的地方或条件下暂停程序。
3. 当程序被停住时，可以检查此时你的程序中所发生的事。
4. 在程序执行过程中修改程序中的变量或条件，将一个bug产生的影响修正从而测试其他bug。

我们这一步需要在qemu上启动我们的i386内核，并用gdb对其进行调试。

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。



The image shows two terminal windows from a VirtualBox environment. The left window displays a kernel panic message: "Kernel panic - not syncing: IO-APIC + timer doesn't work! Boot with apic=debug and send a report. Then try booting with the 'noapic' option." It includes a call trace with timestamps and addresses. The right window shows a GDB session where the user sets a breakpoint at 0xc20db82e in file init/main.c, line 850, and then continues the execution.

```
snow@snow-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[ 0.162615] ..... failed.
[ 0.162765] ...trying to set up timer as Virtual Wire IRQ...
[ 0.179684] ..... failed.
[ 0.179820] ...trying to set up timer as ExtINT IRQ...
[ 2.976248] ..... failed :(
[ 2.976876] Kernel panic - not syncing: IO-APIC + timer doesn't work! Boot w
ith apic=debug and send a report. Then try booting with the 'noapic' option.
[ 2.978939] CPU: 0 PID: 0 Comm: swapper/0 Not tainted 5.10.170 #1
[ 2.979335] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.10.
2-1ubuntu1 04/01/2014
[ 2.980324] Call Trace:
[ 2.981983] dump_stack+0x54/0x68
[ 2.982413] panic+0xb1/0x25a
[ 2.982889] setup_IO_APIC+0x71b/0x76b
[ 2.983426] ? clear_IO_APIC+0x3c/0x60
[ 2.983822] apic_intr_mode_init+0x10e/0x11a
[ 2.984175] x86_late_time_init+0x18/0x29
[ 2.984449] start_kernel+0x39e/0x452
[ 2.984690] i386_start_kernel+0x43/0x45
[ 2.984876] startup_32+0x164/0x168
[ 2.985795] --[ end Kernel panic ]--
! Boot with apic=debug and send a report. Then try booting with the 'noapic' o
ption. ]---

snow@snow-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file linux-5.10.170/vmlinux
Reading symbols from linux-5.10.170/vmlinux...done.
(gdb) target remote:1234
Remote debugging using :1234
0x0000fff0 in ?? ()
(gdb) break strat_kernel
Function "strat_kernel" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) break start_kernel
Breakpoint 1 at 0xc20db82e: file init/main.c, line 850.
(gdb) c
Continuing.
```

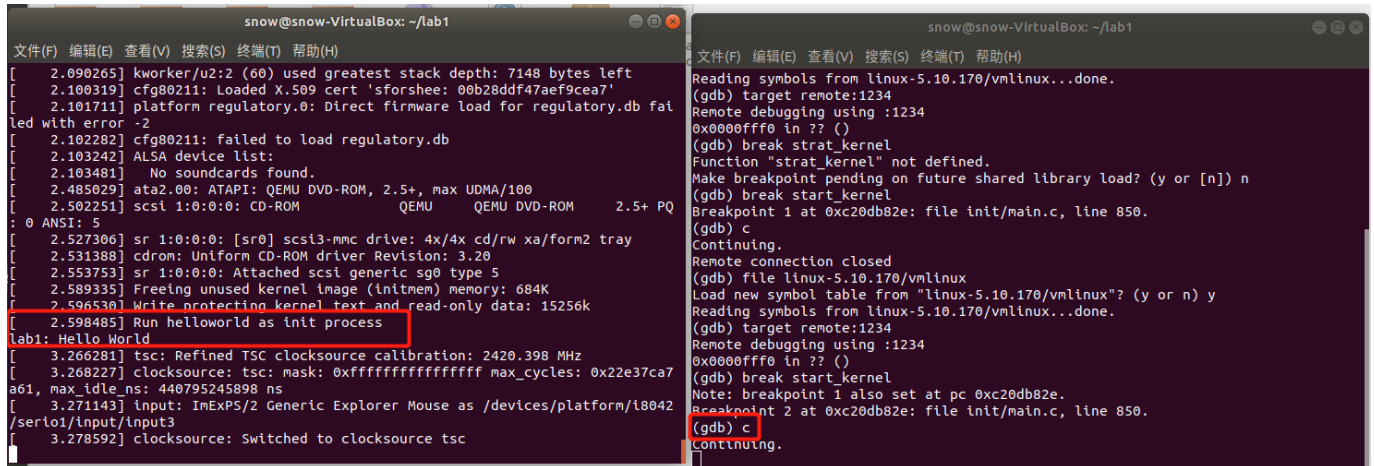
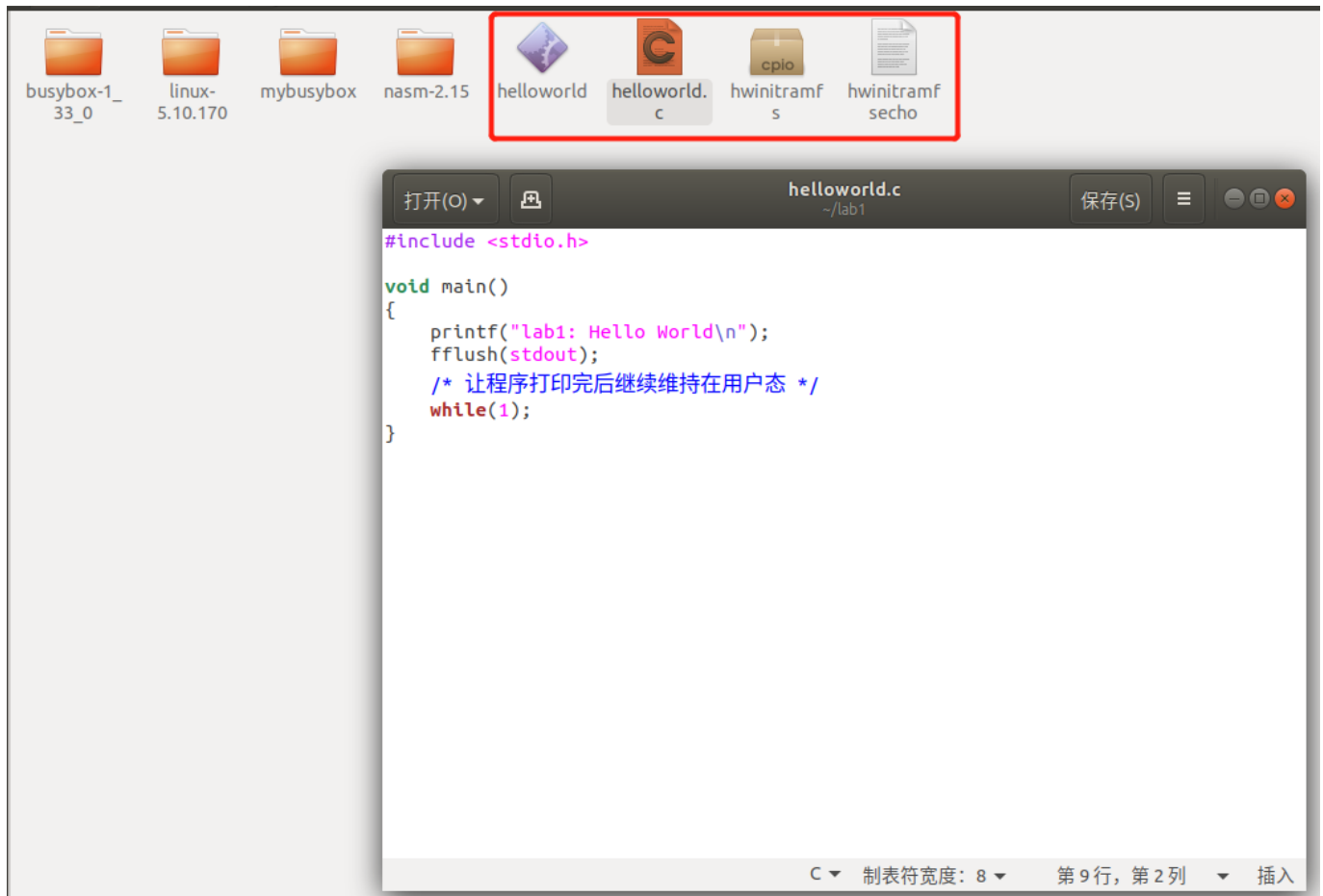
制作Initramfs

这一步是为了解决上一步启动系统的时候只指定了bzImage，没有指定initrd文件，系统无法mount上initrd (init ram disk) 及其initramfs文件系统的问题。

首先了解一下Initramfs是什么

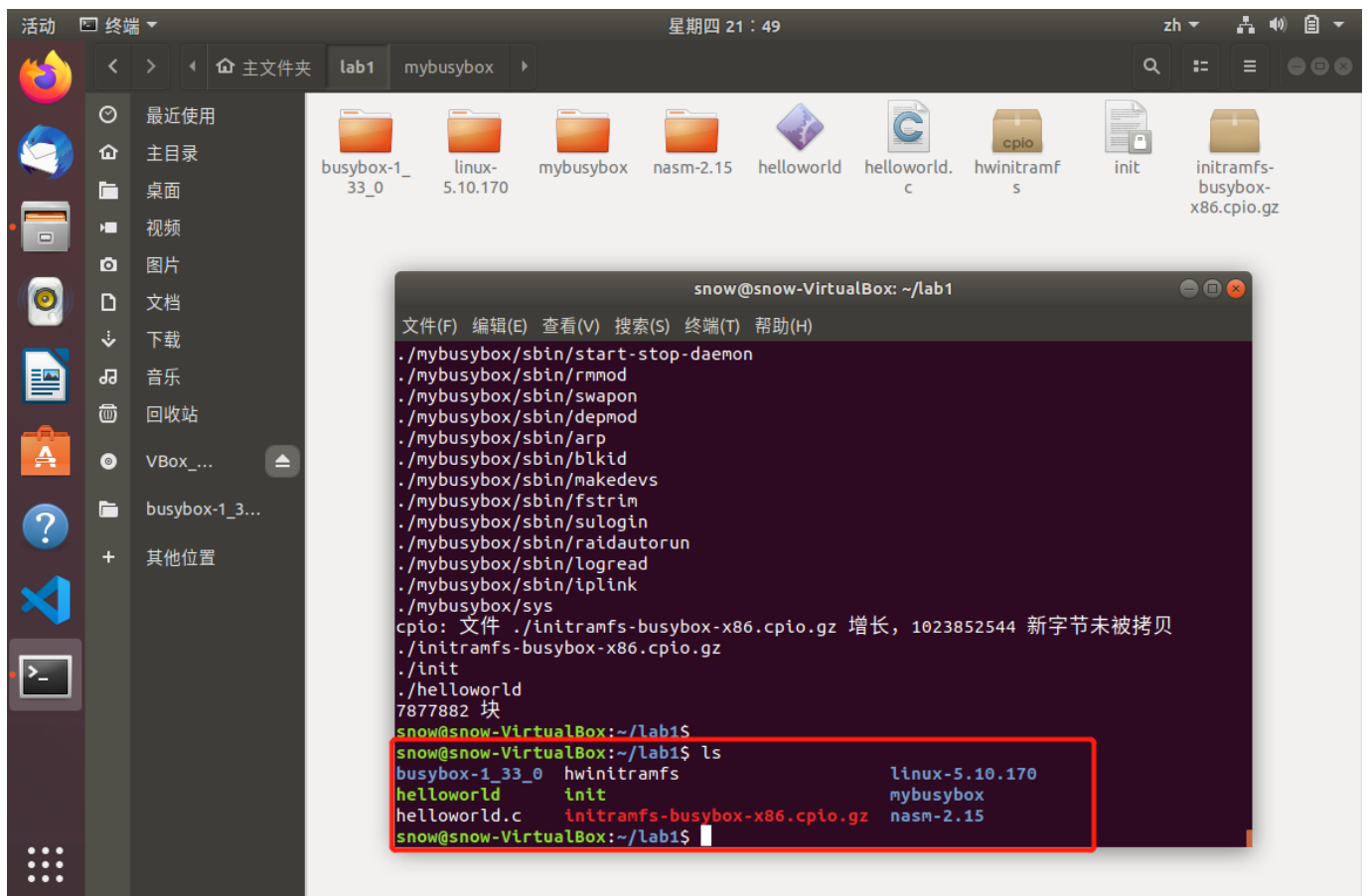
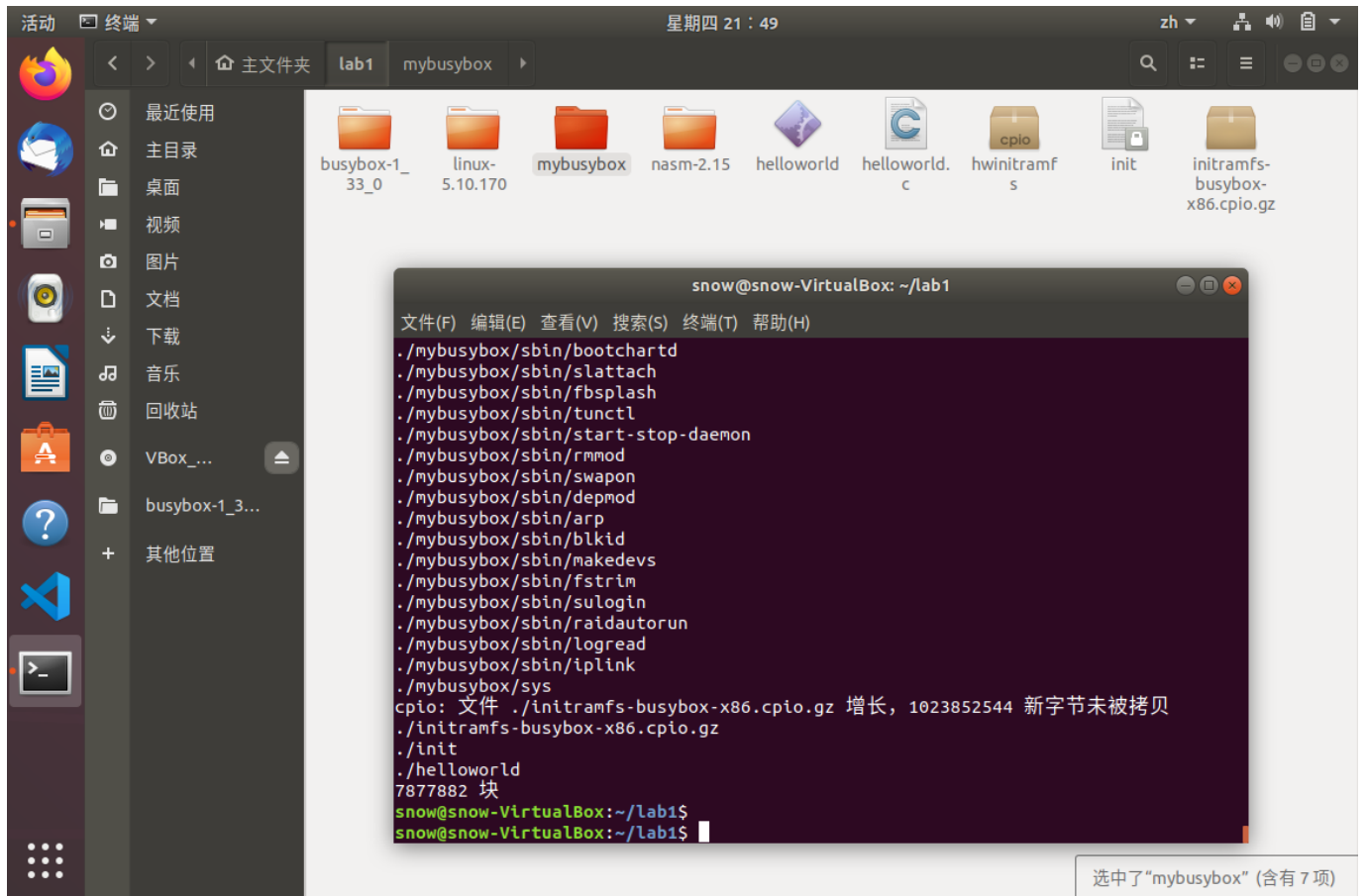
initramfs 即 initram file system，翻译成中文意思就是 **初始 ram 文件系统**，在内核启动的早期提供用一个户态环境，用于完成在内核启动阶段不易完成的工作。

制作好initramfs后，启动内核并且加载，再次启动gdb进行远程调试，可以看到gbd中输出了程序运行的结果。



编译并启动Busybox

busybox是一个开发工具箱，继承压缩了Linux的许多工具和命令，下面是实验结果。



3.关键操作/关键代码

无，这节主要是熟悉一些配置操作。

4.总结

实验中遇到很多不懂的名词，也不太明白这一步的具体步骤为什么要这样做。但是通过查询，最终也大致明白每一步是在做什么。简单来说，主要做的事情就是在Linux环境下通过qemu启动内核并且通过gdb实现了简单的调试，以及配置了必要的工具。

对实验的建议是，希望能在教学文档中增加多一些前置知识（各种名词和工具以及每一步的目的是什么）的解释。

5.参考

[百度百科_全球领先的中文百科全书 \(baidu.com\)](https://baike.baidu.com/)

[在VirtualBox中安装Ubuntu18.04教程_weixin_45841676的博客-CSDN博客](#)

[Linux 命令大全 | 菜鸟教程 \(runoob.com\)](https://runoob.com/)

[程序详细编译过程（预处理、编译、汇编、链接） - 知乎 \(zhihu.com\)](#)

[GCC、GNU到底啥意思？_一只杨阳羊的博客-CSDN博客](#)

[GNU Binutils 介绍 - 会打架的程序员不是好客服 - 博客园 \(cnblogs.com\)](#)

[执行命令：sudo apt-get update时出错，仓库xx不再含有/没有Release文件，无法用该源更新【ubuntu21.04虚拟机】_futureCode的博客-CSDN博客](#)

[Linux / Ubuntu上使用vscode编译运行和调试C/C++ - 知乎 \(zhihu.com\)](#)

[Ubuntu18.04下安装VCode并配置C/C++调试环境_Liu-FE的博客-CSDN博客](#)

[initramfs 在内核中的作用与实现_initramfs是什么_江下枫的博客-CSDN博客](#)