19335030_陈至雪 _ OS第一次实验

实验要求：

- 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。
- 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。
- 实验不限语言， C/C++/Rust都可以。
- 实验不限平台， Windows、Linux和MacOS等都可以。
- 实验不限CPU， ARM/Intel/Risc-V都可以。

实验概述：

1. 搭建OS内核开发环境包括：代码编辑环境、编译环境、运行环境、调试环境等。
2. 下载并编译i386（32位）内核，并利用qemu启动内核。
3. 熟悉制作initramfs的方法。
4. 编写简单应用程序随内核启动运行。
5. 编译i386版本的Busybox，随内核启动，构建简单的OS。
6. 开启远程调试功能，进行调试跟踪代码运行。
7. 撰写实验报告。

实验步骤：

一、环境配置：

1换源：

由于ubuntu的下载源默认是国外的，为了提高下载速度，我们需要将下载源更换为国内源。我们首先备份原先的下载源。

首先备份原先的下载源。

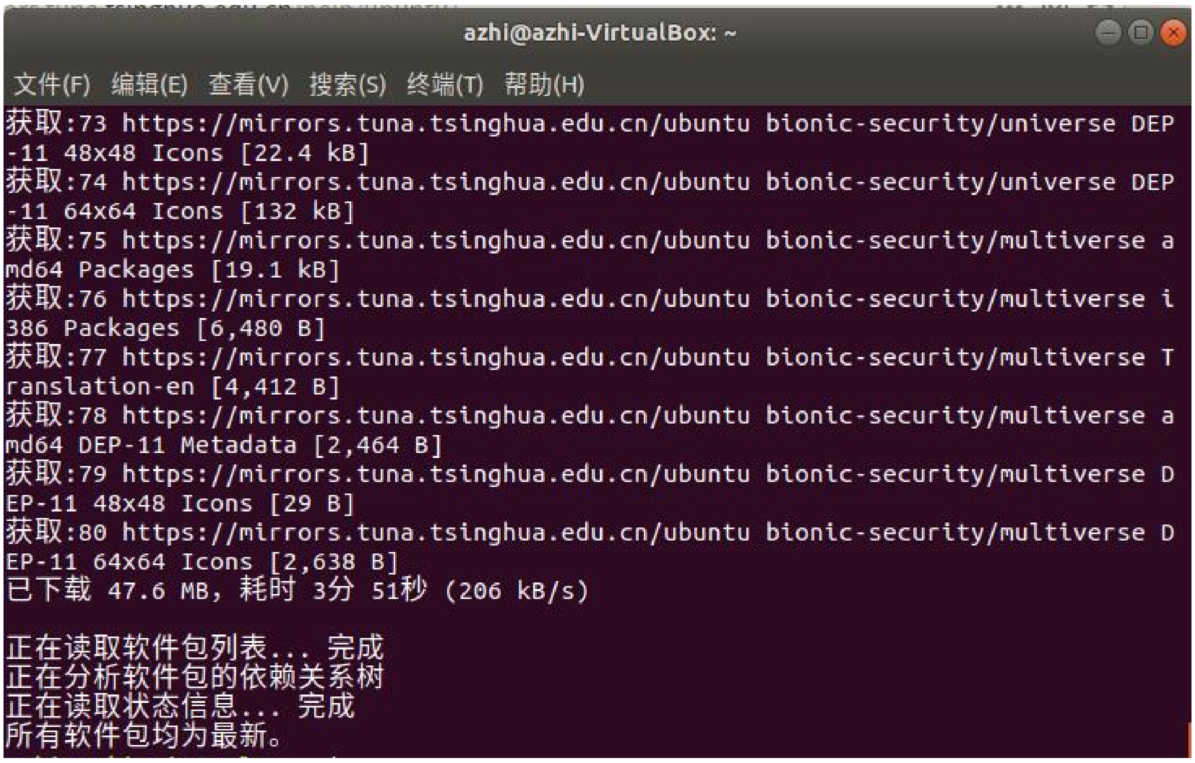
输入命令 `sudo mv /etc/apt/sources.list /etc/apt/sources.list.backup`

然后找到清华的ubuntu下载源[https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu]。注意，选择对应的ubuntu的版本的下载源。

然后使用`gedit`打开下载源保存的文件`/etc/apt/sources.list`/

将下载源复制进`/etc/apt/sources.list`后保存退出。

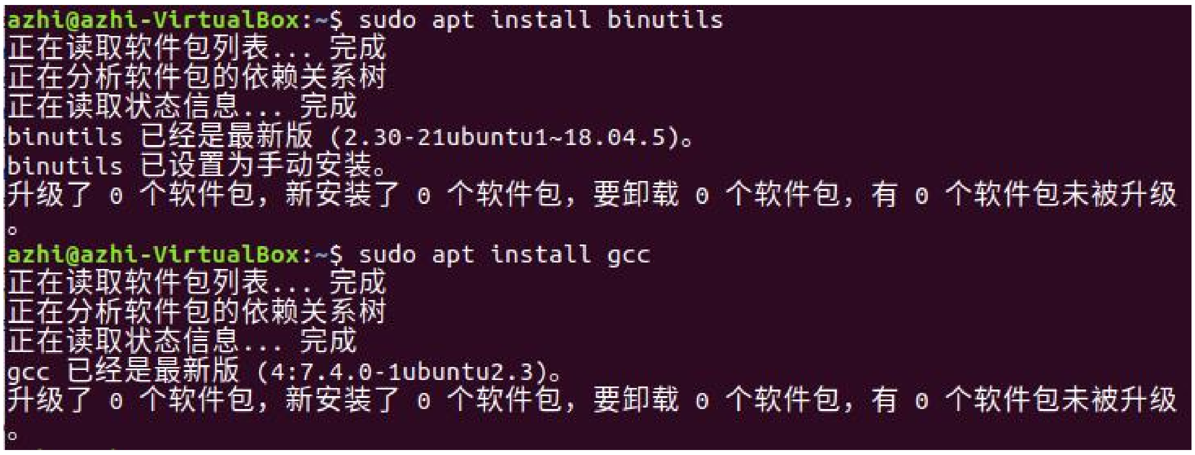
更新apt，检查是否更换成功。输入命令`sudo apt update`查看。成功显示如下：



```
获取:73 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security/universe DEP
-11 48x48 Icons [22.4 kB]
获取:74 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security/universe DEP
-11 64x64 Icons [132 kB]
获取:75 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security/multiverse a
md64 Packages [19.1 kB]
获取:76 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security/multiverse i
386 Packages [6,480 B]
获取:77 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security/multiverse T
ranslation-en [4,412 B]
获取:78 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security/multiverse a
md64 DEP-11 Metadata [2,464 B]
获取:79 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security/multiverse D
EP-11 48x48 Icons [29 B]
获取:80 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security/multiverse D
EP-11 64x64 Icons [2,638 B]
已下载 47.6 MB，耗时 3分 51秒 (206 kB/s)

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
所有软件包均为最新。
```

配置

2、配置C/C++环境



```
azhi@azhi-VirtualBox:~$ sudo apt install binutils
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 (2.30-21ubuntu1~18.04.5)。
binutils 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
。
azhi@azhi-VirtualBox:~$ sudo apt install gcc
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
。
```

3、安装其他工具

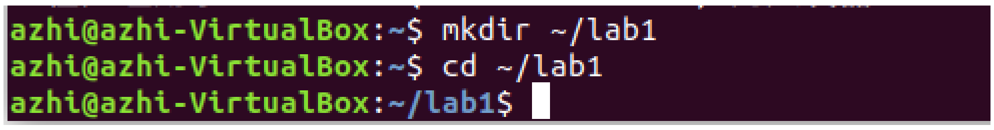
分别输入命令

```
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
```

二、编译Linux内核

1、下载内核：

①在当前用户目录下创建文件夹`lab1`并进入。



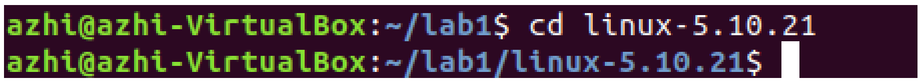
```
azhi@azhi-VirtualBox:~$ mkdir ~/lab1
azhi@azhi-VirtualBox:~$ cd ~/lab1
azhi@azhi-VirtualBox:~/lab1$
```

②到 https://www.kernel.org/ 下载内核5.10到文件夹`~/lab1`。解压并进入，输入命令

`xz -d linux-5.10.20.tar.xz`

`tar -xvf linux-5.10.20`

`cd linux-5.10.20`



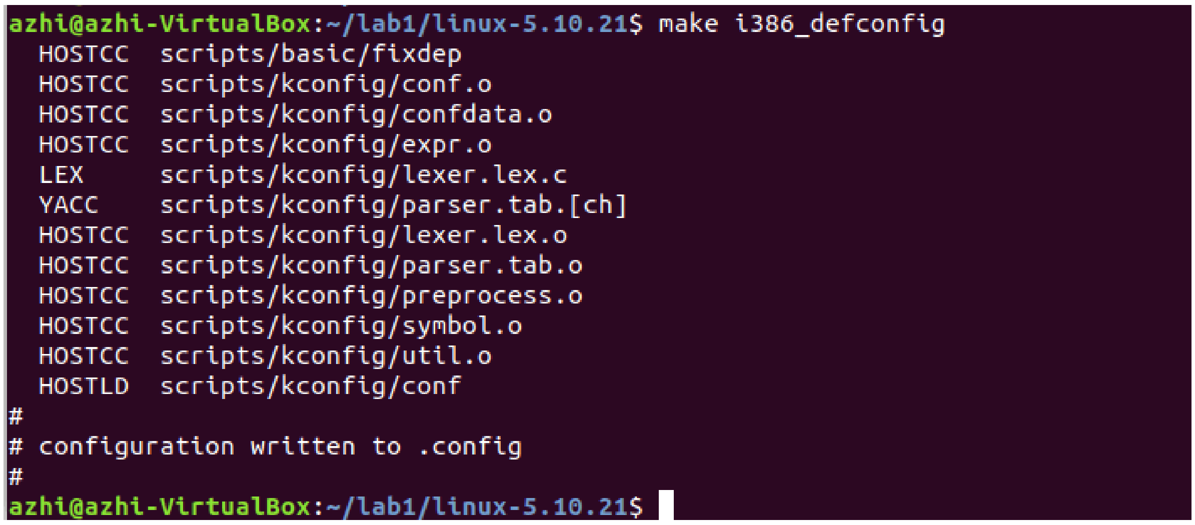
```
azhi@azhi-VirtualBox:~/lab1$ cd linux-5.10.21
azhi@azhi-VirtualBox:~/lab1/linux-5.10.21$
```

2、编译内核：

将内核编译为i386 32位版本。

输入命令

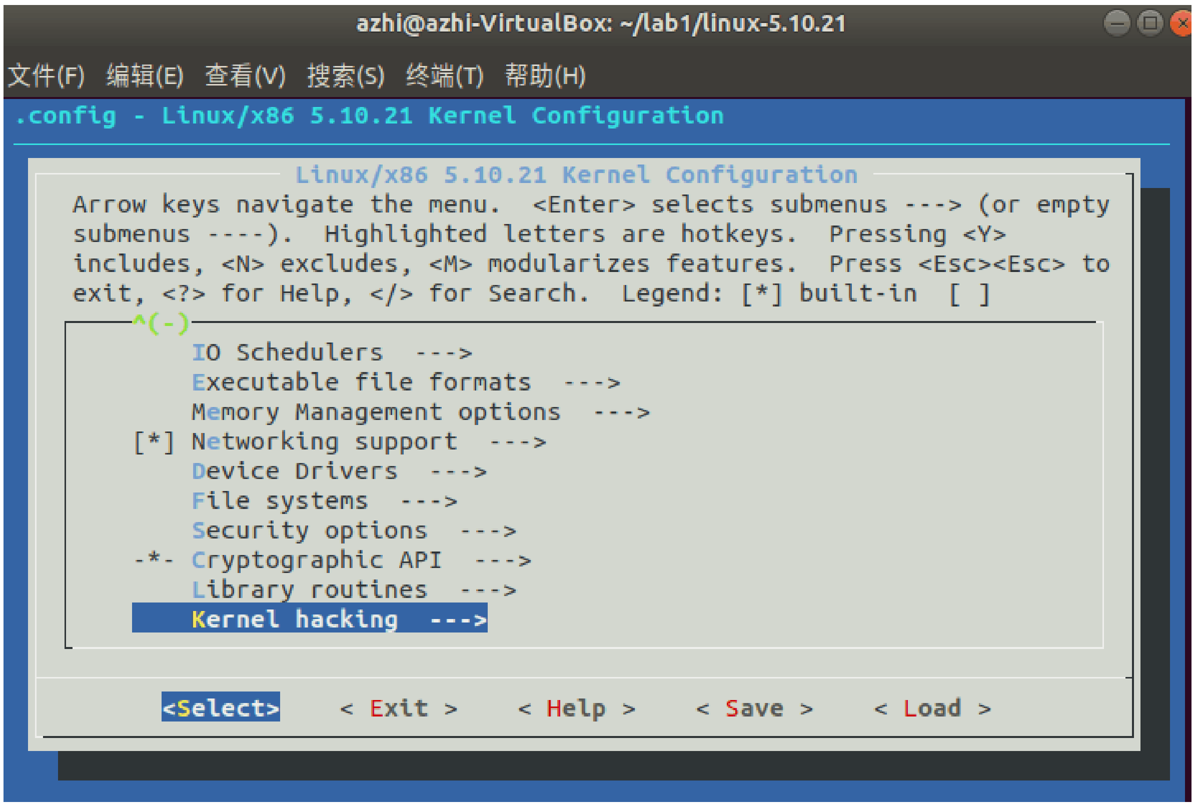
`make i386_defconfig`



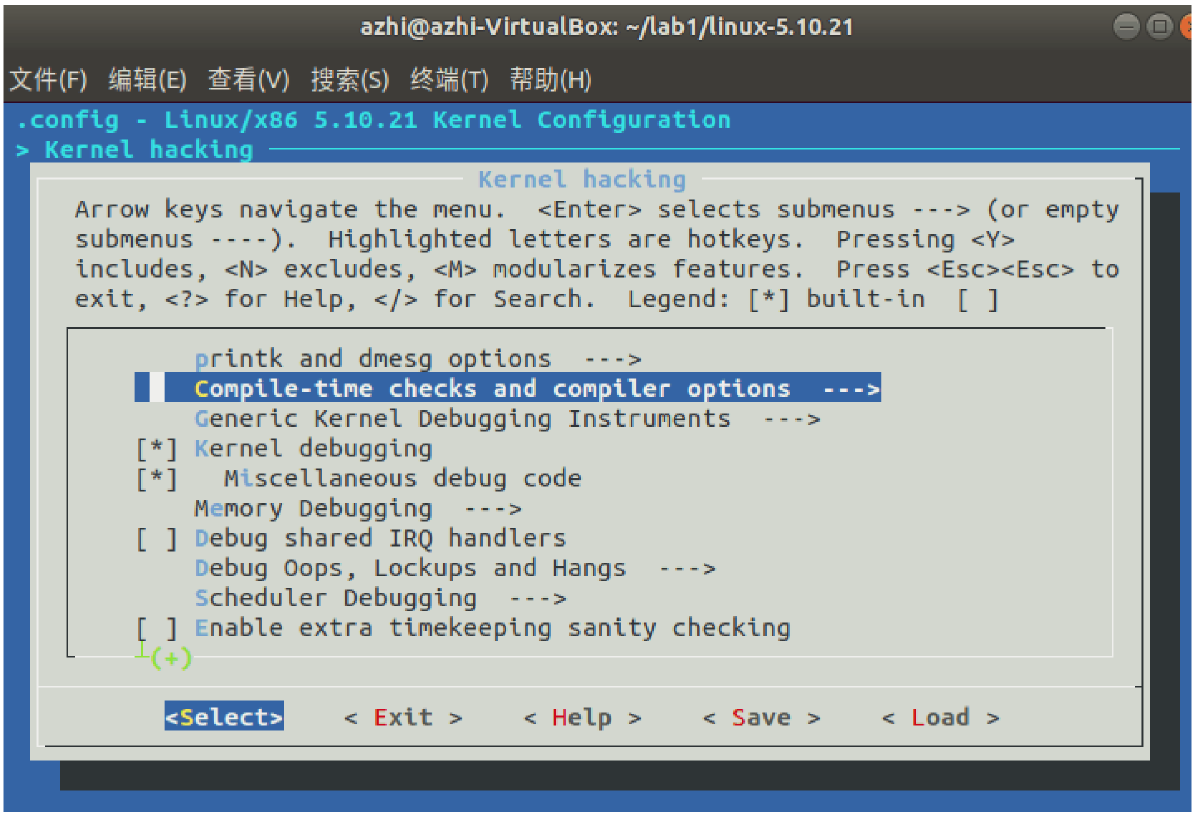
```
azhi@azhi-VirtualBox:~/lab1/linux-5.10.21$ make i386_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTCC  scripts/kconfig/util.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
azhi@azhi-VirtualBox:~/lab1/linux-5.10.21$
```

输入命令`make menuconfig`

在打开的图像界面中依次选择`Kernel hacking`、`Compile-time checks and compiler options`，最后在`[ ] Compile the kernel with debug info`输入`Y`勾选，保存退出。


azhi@azhi-VirtualBox: ~/lab1/linux-5.10.21
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

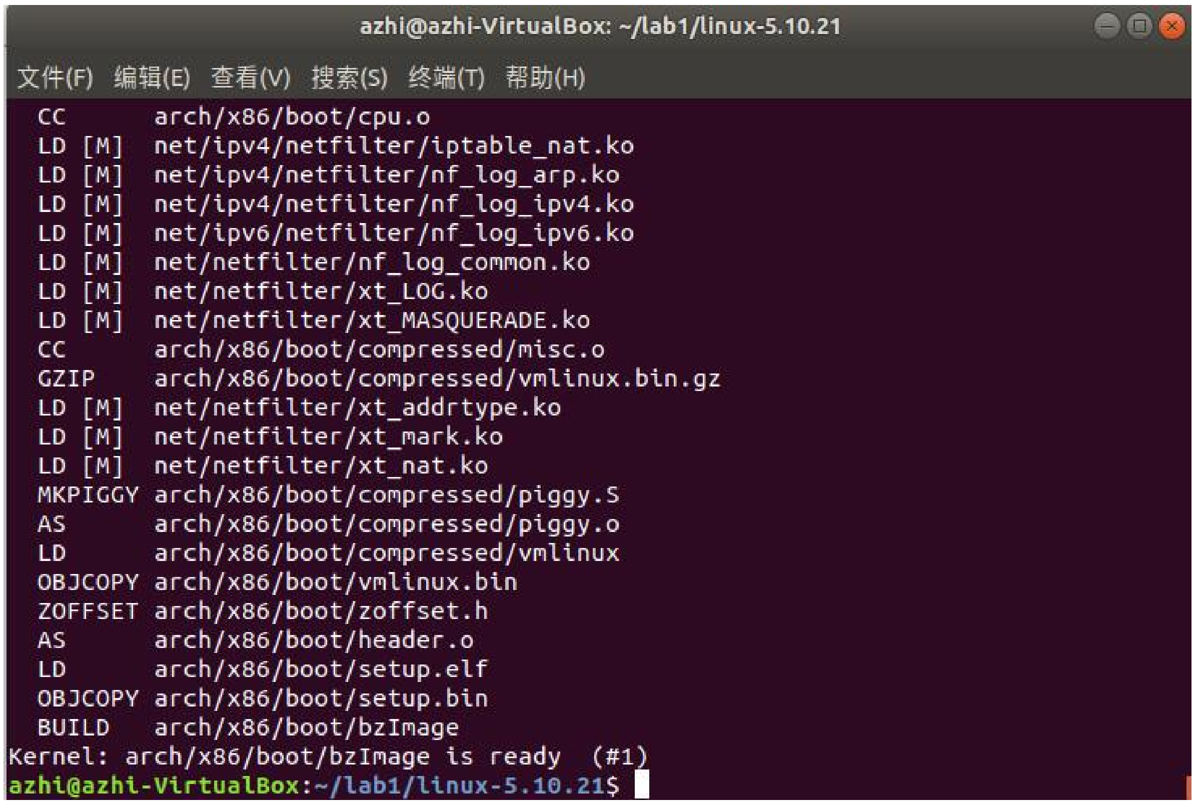
```
.config - Linux/x86 5.10.21 Kernel Configuration
                 Linux/x86 5.10.21 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
      ^(-)
            IO Schedulers  --->
            Executable file formats  --->
            Memory Management options  --->
        [*] Networking support  --->
            Device Drivers  --->
            File systems  --->
            Security options  --->
        -*- Cryptographic API  --->
            Library routines  --->
            Kernel hacking  --->

        <Select>    < Exit >    < Help >    < Save >    < Load >
```


azhi@azhi-VirtualBox: ~/lab1/linux-5.10.21
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
.config - Linux/x86 5.10.21 Kernel Configuration
> Kernel hacking
                          Kernel hacking
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

            printk and dmesg options  --->
            Compile-time checks and compiler options  --->
            Generic Kernel Debugging Instruments  --->
        [*] Kernel debugging
        [*]   Miscellaneous debug code
            Memory Debugging  --->
        [ ] Debug shared IRQ handlers
            Debug Oops, Lockups and Hangs  --->
            Scheduler Debugging  --->
        [ ] Enable extra timekeeping sanity checking
      ↓(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

编译内核，输入命令`make -j8`

检查Linux压缩镜像`linux-5.10.21/arch/x86/boot/bzImage`和符号表`linux-5.10.21/vmlinux`是否已经生成。


azhi@azhi-VirtualBox: ~/lab1/linux-5.10.21
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
  CC      arch/x86/boot/cpu.o
  LD [M]  net/ipv4/netfilter/iptable_nat.ko
  LD [M]  net/ipv4/netfilter/nf_log_arp.ko
  LD [M]  net/ipv4/netfilter/nf_log_ipv4.ko
  LD [M]  net/ipv6/netfilter/nf_log_ipv6.ko
  LD [M]  net/netfilter/nf_log_common.ko
  LD [M]  net/netfilter/xt_LOG.ko
  LD [M]  net/netfilter/xt_MASQUERADE.ko
  CC      arch/x86/boot/compressed/misc.o
  GZIP    arch/x86/boot/compressed/vmlinux.bin.gz
  LD [M]  net/netfilter/xt_addrtype.ko
  LD [M]  net/netfilter/xt_mark.ko
  LD [M]  net/netfilter/xt_nat.ko
  MKPIGGY arch/x86/boot/compressed/piggy.S
  AS      arch/x86/boot/compressed/piggy.o
  LD      arch/x86/boot/compressed/vmlinux
  OBJCOPY arch/x86/boot/vmlinux.bin
  ZOFFSET arch/x86/boot/zoffset.h
  AS      arch/x86/boot/header.o
  LD      arch/x86/boot/setup.elf
  OBJCOPY arch/x86/boot/setup.bin
  BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready  (#1)
azhi@azhi-VirtualBox:~/lab1/linux-5.10.21$
```

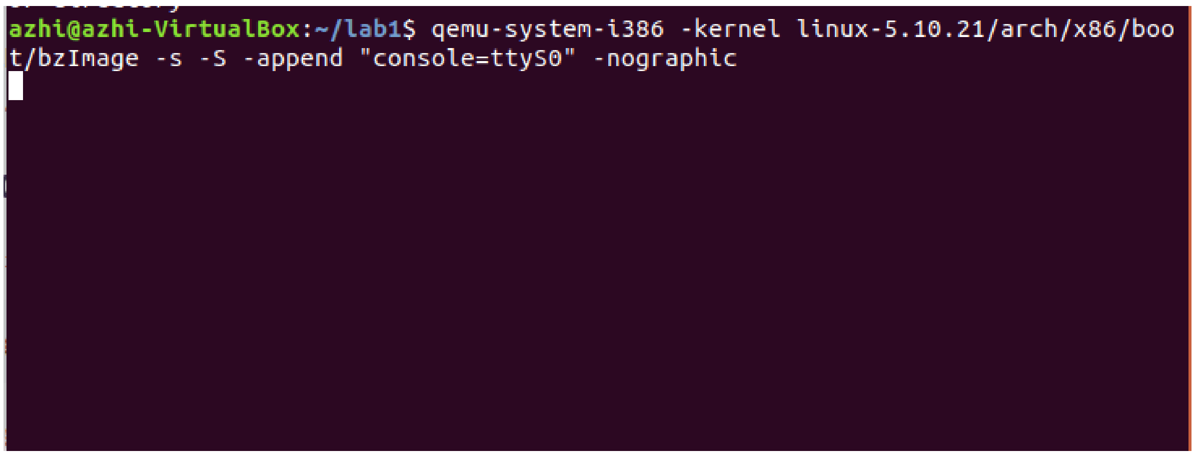
可以看到Linux压缩镜像和符号表已经生成。

三、启动内核并调试

1、启动qemu

使用`qemu`启动内核并开启远程调试。

在lab1下输入命令`qemu-system-i386 -kernel linux-5.10.21/arch/x86/boot/bzImage -s -S -append "console=ttyS0" -nographic`



```
azhi@azhi-VirtualBox:~/lab1$ qemu-system-i386 -kernel linux-5.10.21/arch/x86/boo
t/bzImage -s -S -append "console=ttyS0" -nographic
```

这里，qemu并没有输出任何信息。qemu在等待gdb输入指令。

接下来启动gdb，通过gdb指令告诉qemu怎么做。

2、gdb调试

在另一个终端下启动gdb，注意不要关闭qemu所在的终端。

进入目录lab1，输入命令：

`cd ~/lab1`

进入gdb，输入`gdb`

在gdb下加载符号表：

`file linux-5.10.21/vmlinux`

在gdb下，连接已经启动的qemu进行调试。

```
target remote:1234
```

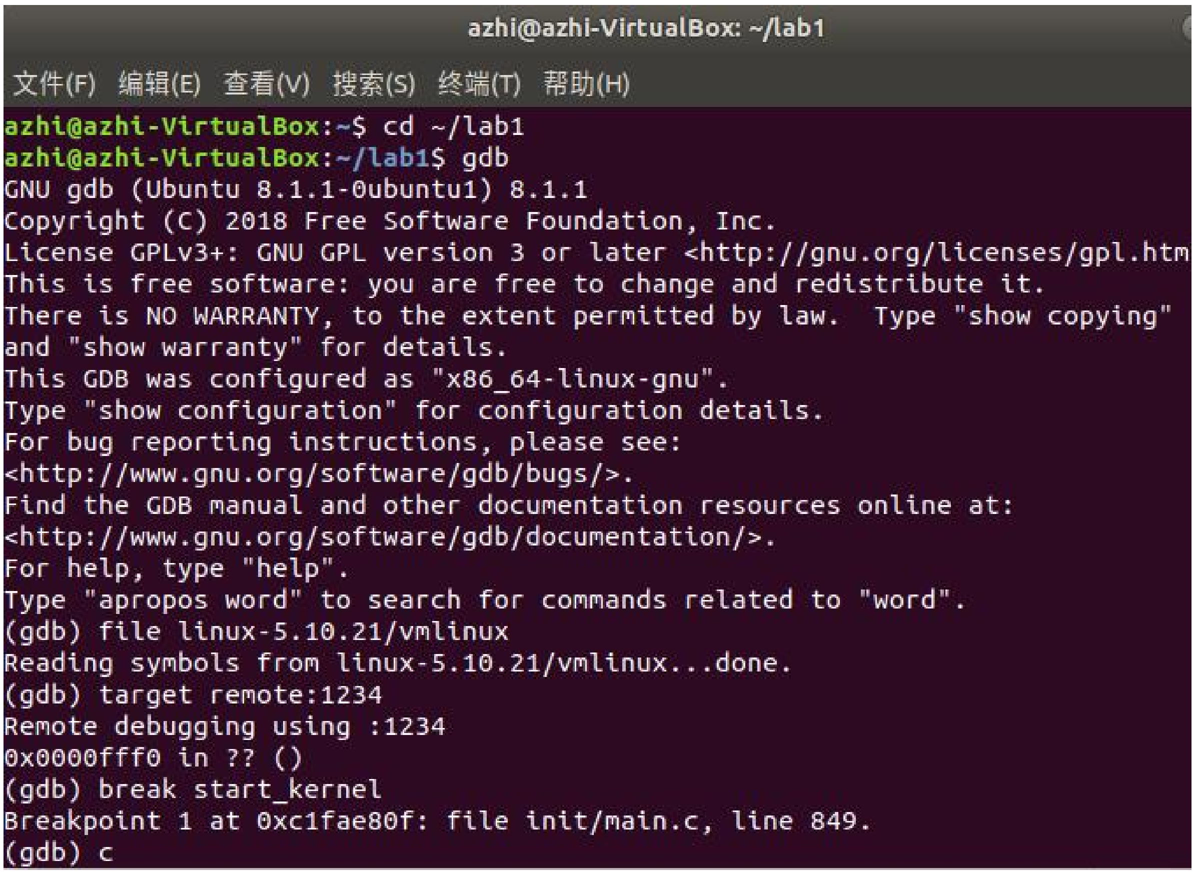
在gdb下，为start_kernel函数设置断点。

```
break start_kernel
```

在gdb下，输入`c`运行。

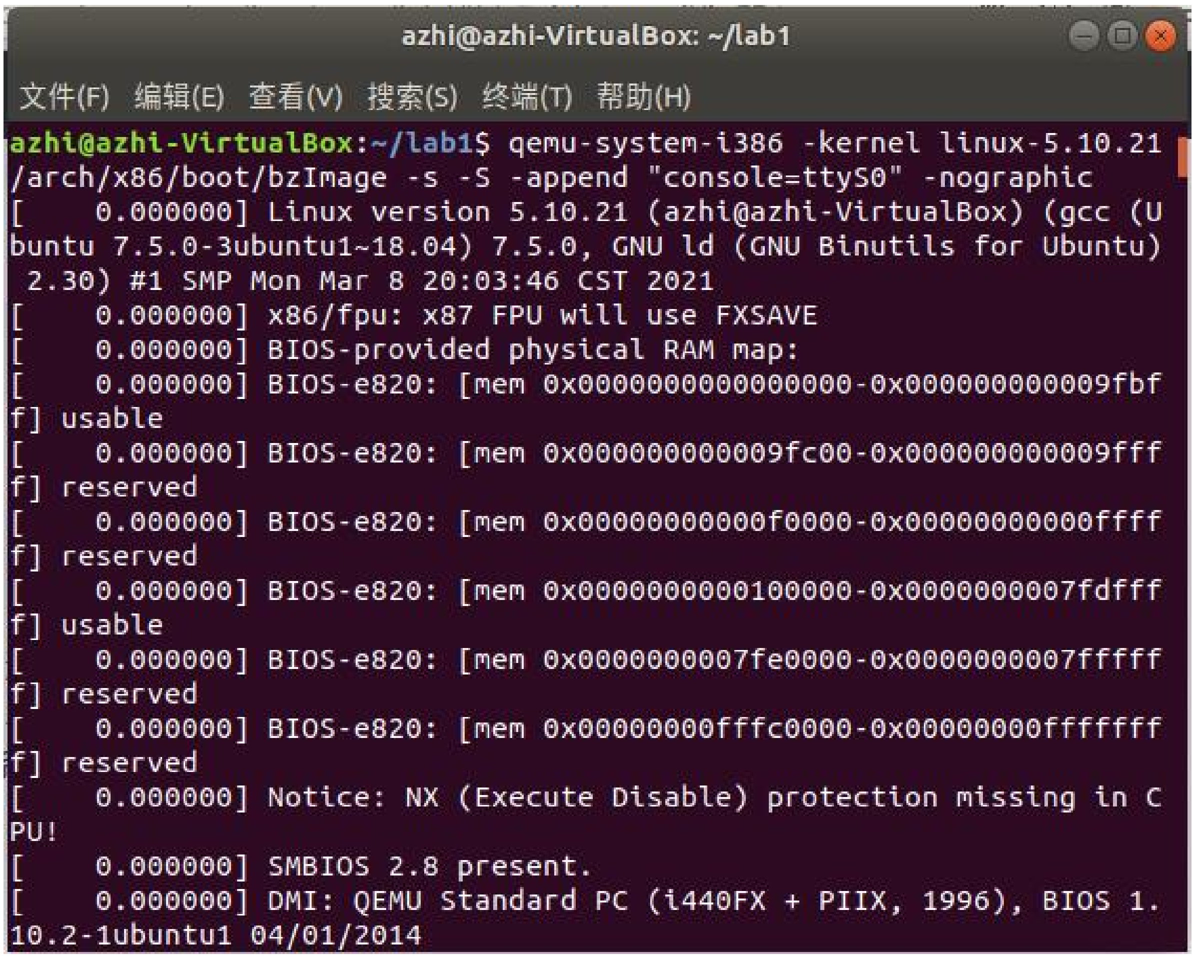
```
c
```

如图示：

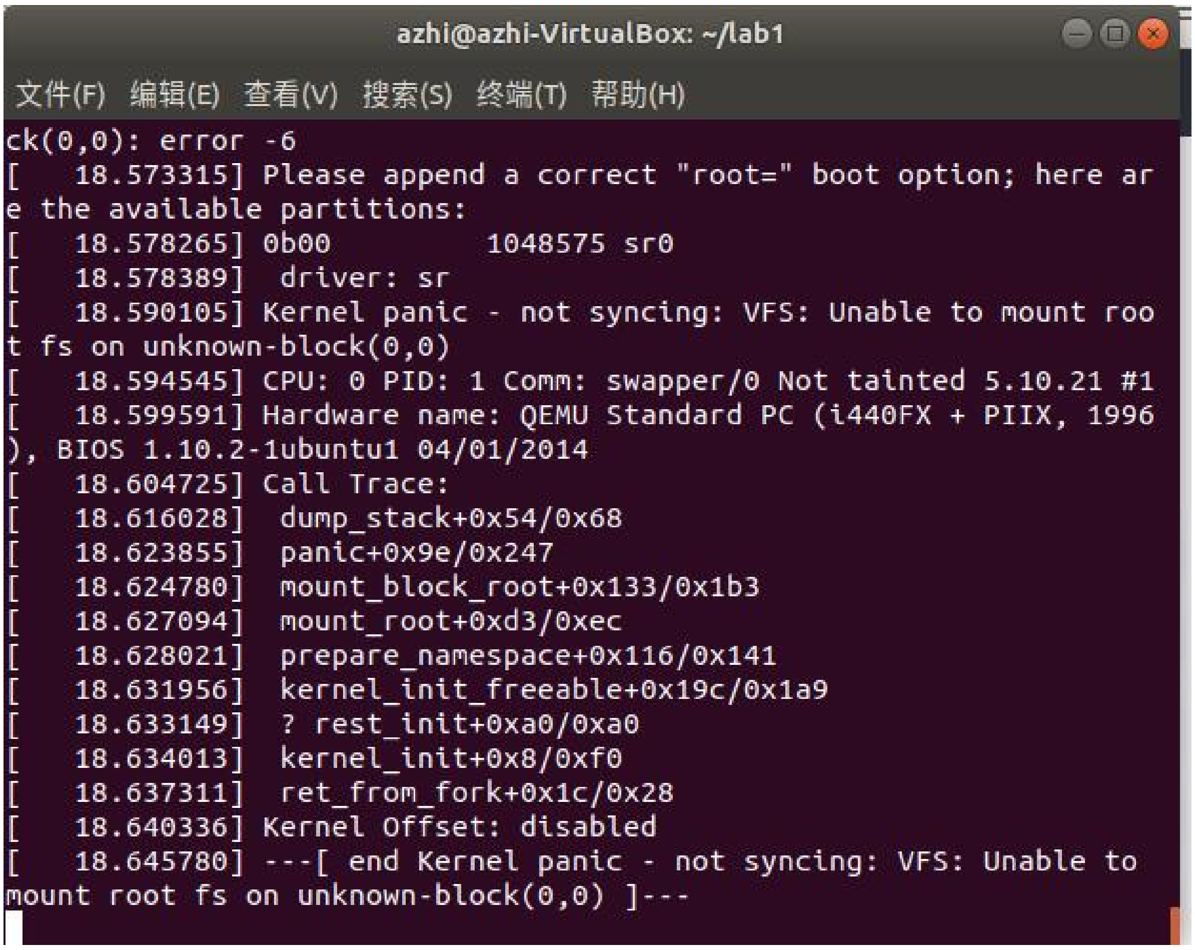

azhi@azhi-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
azhi@azhi-VirtualBox:~$ cd ~/lab1
azhi@azhi-VirtualBox:~/lab1$ gdb
GNU gdb (Ubuntu 8.1.1-0ubuntu1) 8.1.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
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-5.10.21/vmlinux
Reading symbols from linux-5.10.21/vmlinux...done.
(gdb) target remote:1234
Remote debugging using :1234
0x0000fff0 in ?? ()
(gdb) break start_kernel
Breakpoint 1 at 0xc1fae80f: file init/main.c, line 849.
(gdb) c
```

在继续执行后，最终qemu的输出如下


azhi@azhi-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
azhi@azhi-VirtualBox:~/lab1$ qemu-system-i386 -kernel linux-5.10.21
/arch/x86/boot/bzImage -s -S -append "console=ttyS0" -nographic
[    0.000000] Linux version 5.10.21 (azhi@azhi-VirtualBox) (gcc (U
buntu 7.5.0-3ubuntu1~18.04) 7.5.0, GNU ld (GNU Binutils for Ubuntu)
 2.30) #1 SMP Mon Mar 8 20:03:46 CST 2021
[    0.000000] x86/fpu: x87 FPU will use FXSAVE
[    0.000000] BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbf
f] usable
[    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009fff
f] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000ffff
f] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x0000000007fdfff
f] usable
[    0.000000] BIOS-e820: [mem 0x0000000007fe0000-0x0000000007fffff
f] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fffc0000-0x00000000fffffff
f] reserved
[    0.000000] Notice: NX (Execute Disable) protection missing in C
PU!
[    0.000000] SMBIOS 2.8 present.
[    0.000000] DMI: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.
10.2-1ubuntu1 04/01/2014
```


azhi@azhi-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
ck(0,0): error -6
[   18.573315] Please append a correct "root=" boot option; here ar
e the available partitions:
[   18.578265] 0b00         1048575 sr0
[   18.578389]  driver: sr
[   18.590105] Kernel panic - not syncing: VFS: Unable to mount roo
t fs on unknown-block(0,0)
[   18.594545] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 5.10.21 #1
[   18.599591] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996
), BIOS 1.10.2-1ubuntu1 04/01/2014
[   18.604725] Call Trace:
[   18.616028]  dump_stack+0x54/0x68
[   18.623855]  panic+0x9e/0x247
[   18.624780]  mount_block_root+0x133/0x1b3
[   18.627094]  mount_root+0xd3/0xec
[   18.628021]  prepare_namespace+0x116/0x141
[   18.631956]  kernel_init_freeable+0x19c/0x1a9
[   18.633149]  ? rest_init+0xa0/0xa0
[   18.634013]  kernel_init+0x8/0xf0
[   18.637311]  ret_from_fork+0x1c/0x28
[   18.640336] Kernel Offset: disabled
[   18.645780] ---[ end Kernel panic - not syncing: VFS: Unable to
mount root fs on unknown-block(0,0) ]---
```

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。

四、制作Initramfs

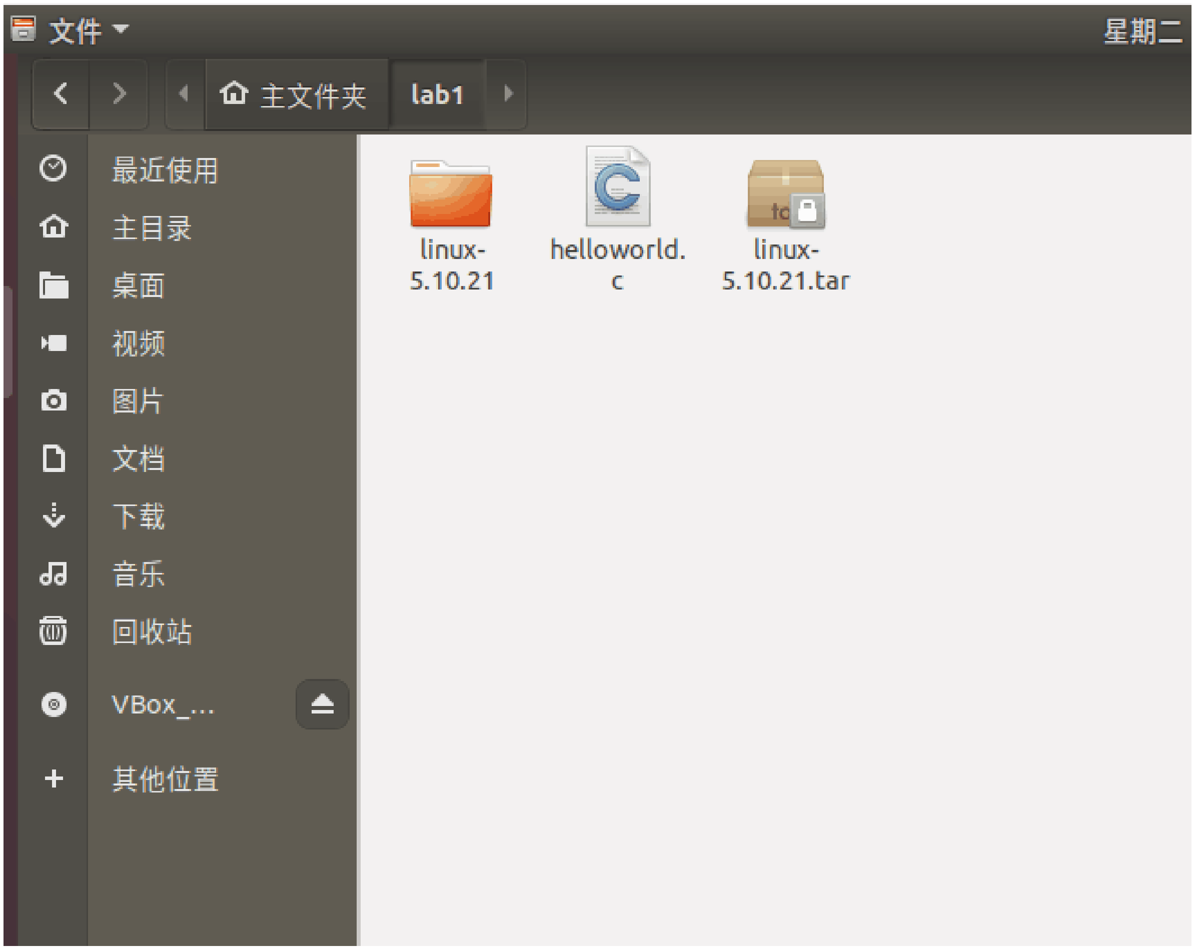
首先`cd ~/lab1`

1、Hello World

做一个最简单的Hello World initramfs，来直观地理解initramfs，Hello World. 程序如下:

```
#include <stdio.h>
void main()
{
    printf("lab1: Hello world\n");
    fflush(stdout);
    /* 让程序打印完后继续维持在用户态 */
    while(1);
}
```

上述文件保存在`~/lab1/helloworld.c`中，



然后将上面代码编译成32位可执行文件，输入命令

```
gcc -o helloworld -m32 -static helloworld.c
```

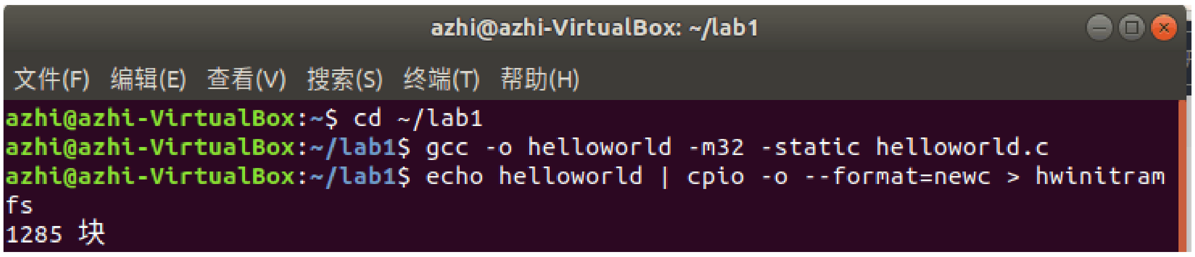
2、加载initramfs

用cpio打包initramfs。

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

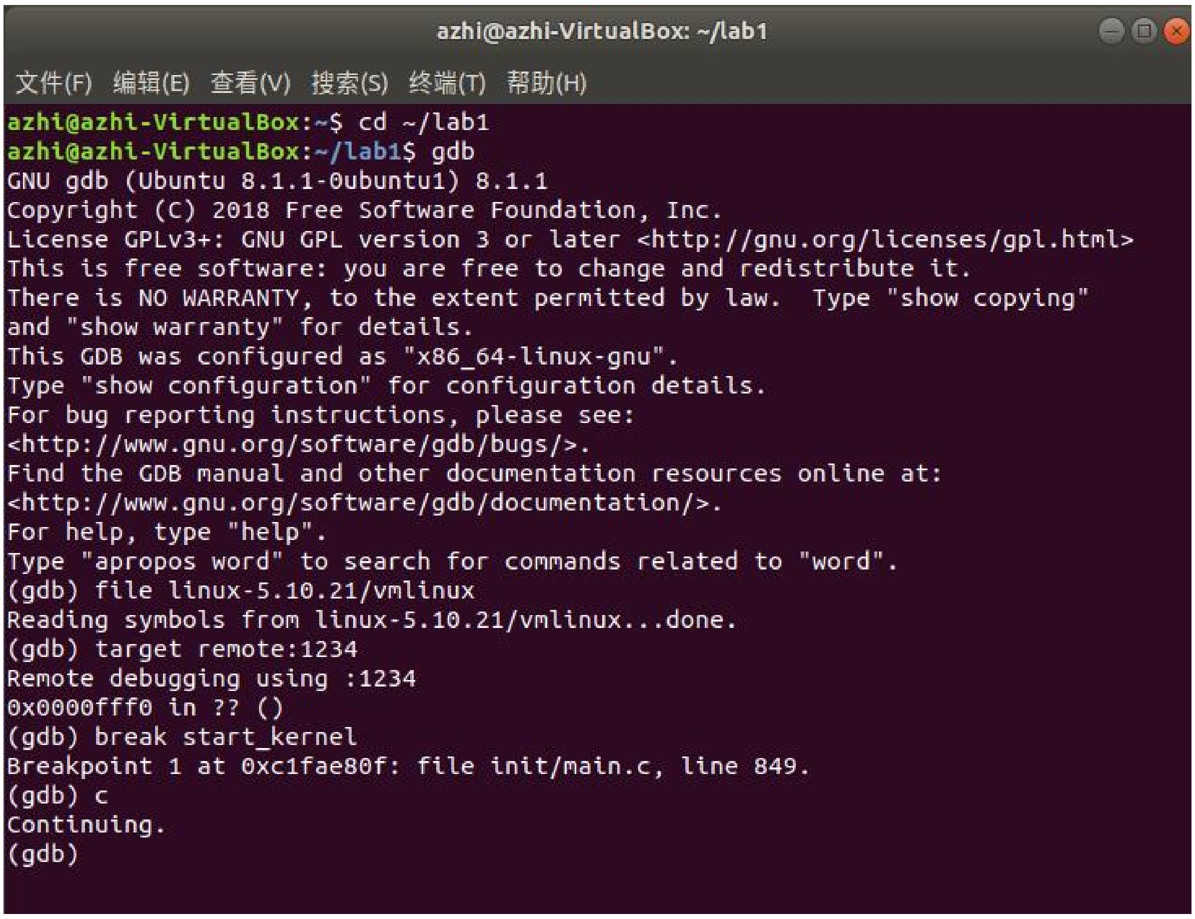
启动内核，并加载initramfs。

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


azhi@azhi-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

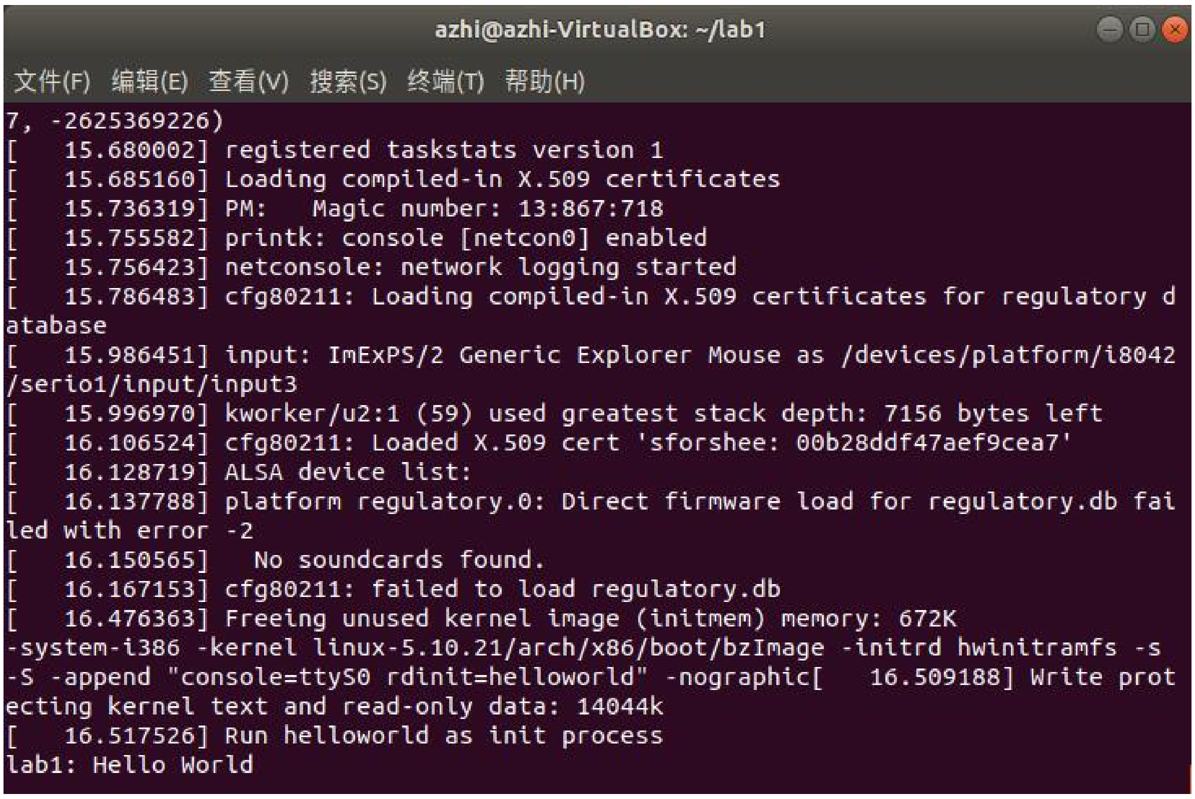
```
azhi@azhi-VirtualBox:~$ cd ~/lab1
azhi@azhi-VirtualBox:~/lab1$ gcc -o helloworld -m32 -static helloworld.c
azhi@azhi-VirtualBox:~/lab1$ echo helloworld | cpio -o --format=newc > hwinitram
fs
1285 块
```

然后重复一遍上面的gdb调试过程，


azhi@azhi-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
azhi@azhi-VirtualBox:~$ cd ~/lab1
azhi@azhi-VirtualBox:~/lab1$ gdb
GNU gdb (Ubuntu 8.1.1-0ubuntu1) 8.1.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
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-5.10.21/vmlinux
Reading symbols from linux-5.10.21/vmlinux...done.
(gdb) target remote:1234
Remote debugging using :1234
0x0000fff0 in ?? ()
(gdb) break start_kernel
Breakpoint 1 at 0xc1fae80f: file init/main.c, line 849.
(gdb) c
Continuing.
(gdb)
```

可以看到`lab1: Hello world\n`,如下图


azhi@azhi-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
7, -2625369226)
[   15.680002] registered taskstats version 1
[   15.685160] Loading compiled-in X.509 certificates
[   15.736319] PM:   Magic number: 13:867:718
[   15.755582] printk: console [netcon0] enabled
[   15.756423] netconsole: network logging started
[   15.786483] cfg80211: Loading compiled-in X.509 certificates for regulatory d
atabase
[   15.986451] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i8042
/serio1/input/input3
[   15.996970] kworker/u2:1 (59) used greatest stack depth: 7156 bytes left
[   16.106524] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[   16.128719] ALSA device list:
[   16.137788] platform regulatory.0: Direct firmware load for regulatory.db fai
led with error -2
[   16.150565]   No soundcards found.
[   16.167153] cfg80211: failed to load regulatory.db
[   16.476363] Freeing unused kernel image (initmem) memory: 672K
-system-i386 -kernel linux-5.10.21/arch/x86/boot/bzImage -initrd hwinitramfs -s
-S -append "console=ttyS0 rdinit=helloworld" -nographic[   16.509188] Write prot
ecting kernel text and read-only data: 14044k
[   16.517526] Run helloworld as init process
lab1: Hello World
```

五、编译并启动Busybox

1、下载并解压

下载Busybox压缩包到目录lab1下，然后解压。

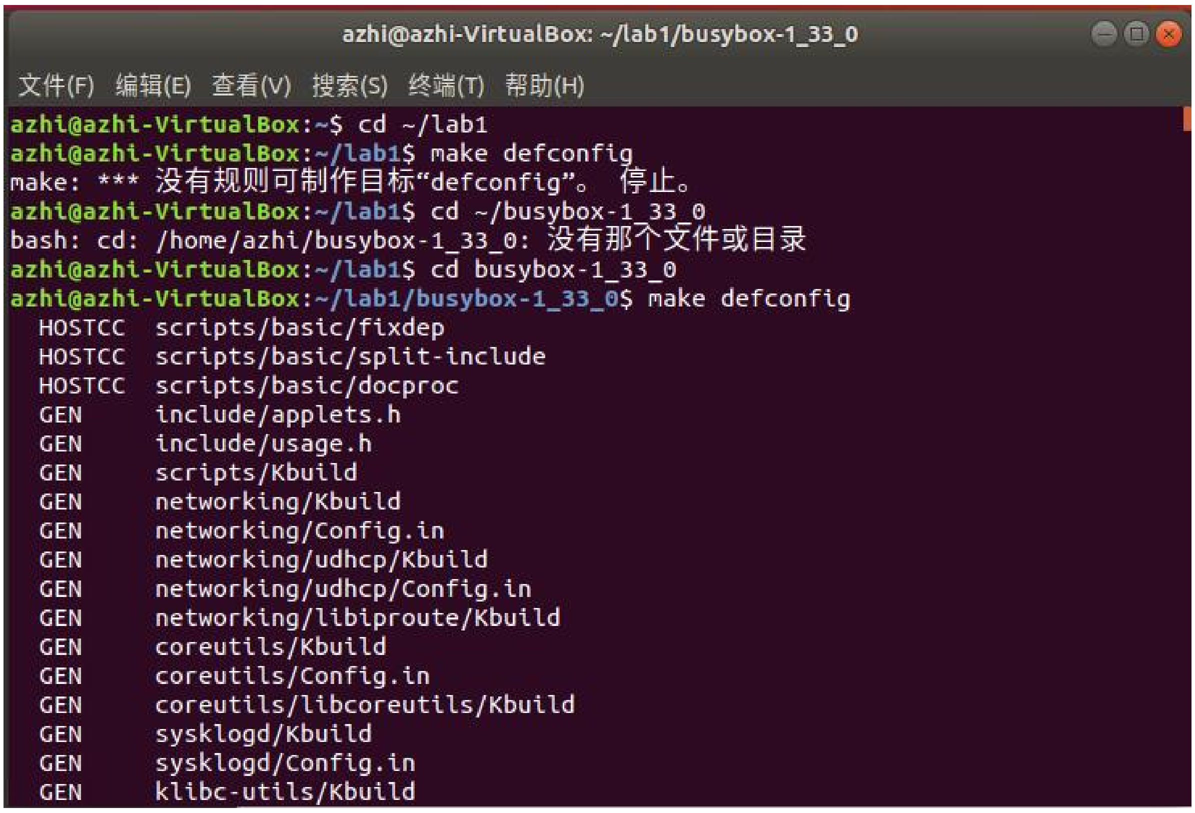
先进入到~/lab1,`cd ~/lab1`

然后解压`tar -xf Busybox`

2、编译busybox

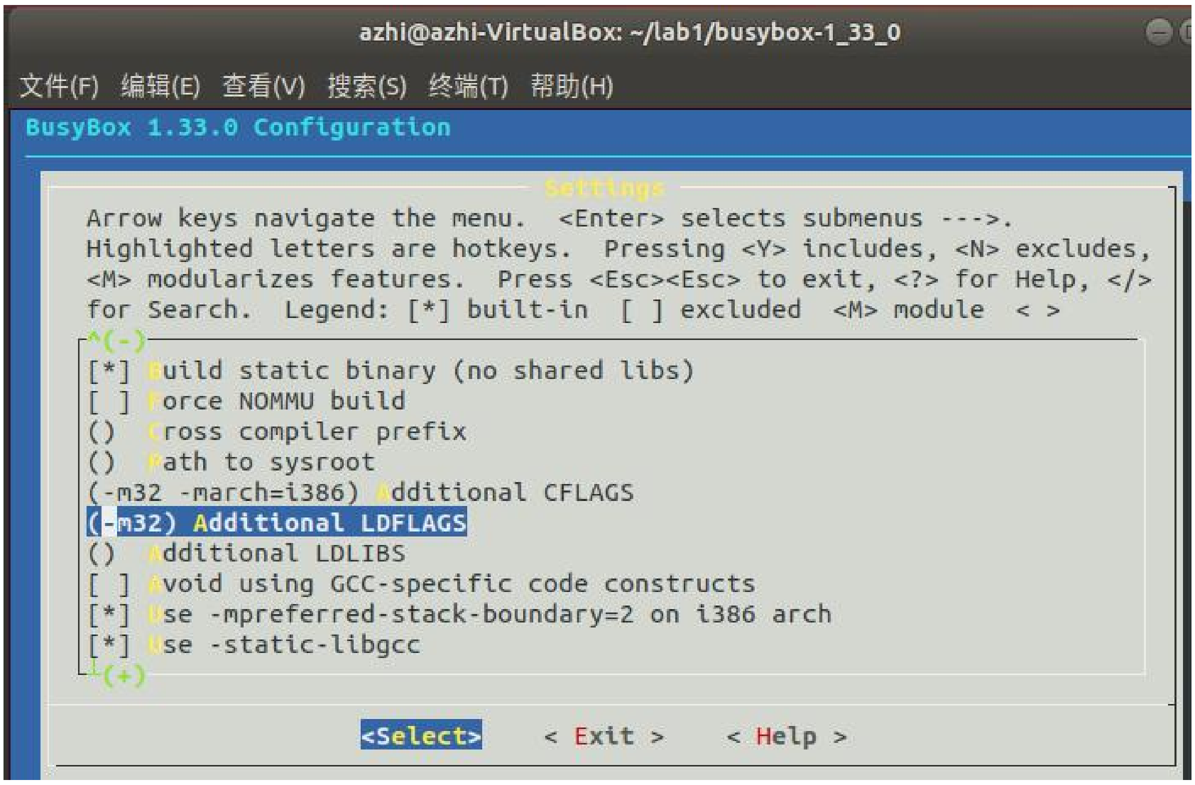
先进入Busybox_1_33_0目录再进行操作：

```
make defconfig
make menuconfig
```



```
azhi@azhi-VirtualBox:~$ cd ~/lab1
azhi@azhi-VirtualBox:~/lab1$ make defconfig
make: *** 没有规则可制作目标“defconfig”。 停止。
azhi@azhi-VirtualBox:~/lab1$ cd ~/busybox-1_33_0
bash: cd: /home/azhi/busybox-1_33_0: 没有那个文件或目录
azhi@azhi-VirtualBox:~/lab1$ cd busybox-1_33_0
azhi@azhi-VirtualBox:~/lab1/busybox-1_33_0$ make defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/basic/split-include
  HOSTCC  scripts/basic/docproc
  GEN     include/applets.h
  GEN     include/usage.h
  GEN     scripts/Kbuild
  GEN     networking/Kbuild
  GEN     networking/Config.in
  GEN     networking/udhcp/Kbuild
  GEN     networking/udhcp/Config.in
  GEN     networking/libiproute/Kbuild
  GEN     coreutils/Kbuild
  GEN     coreutils/Config.in
  GEN     coreutils/libcoreutils/Kbuild
  GEN     sysklogd/Kbuild
  GEN     sysklogd/Config.in
  GEN     klibc-utils/Kbuild
```

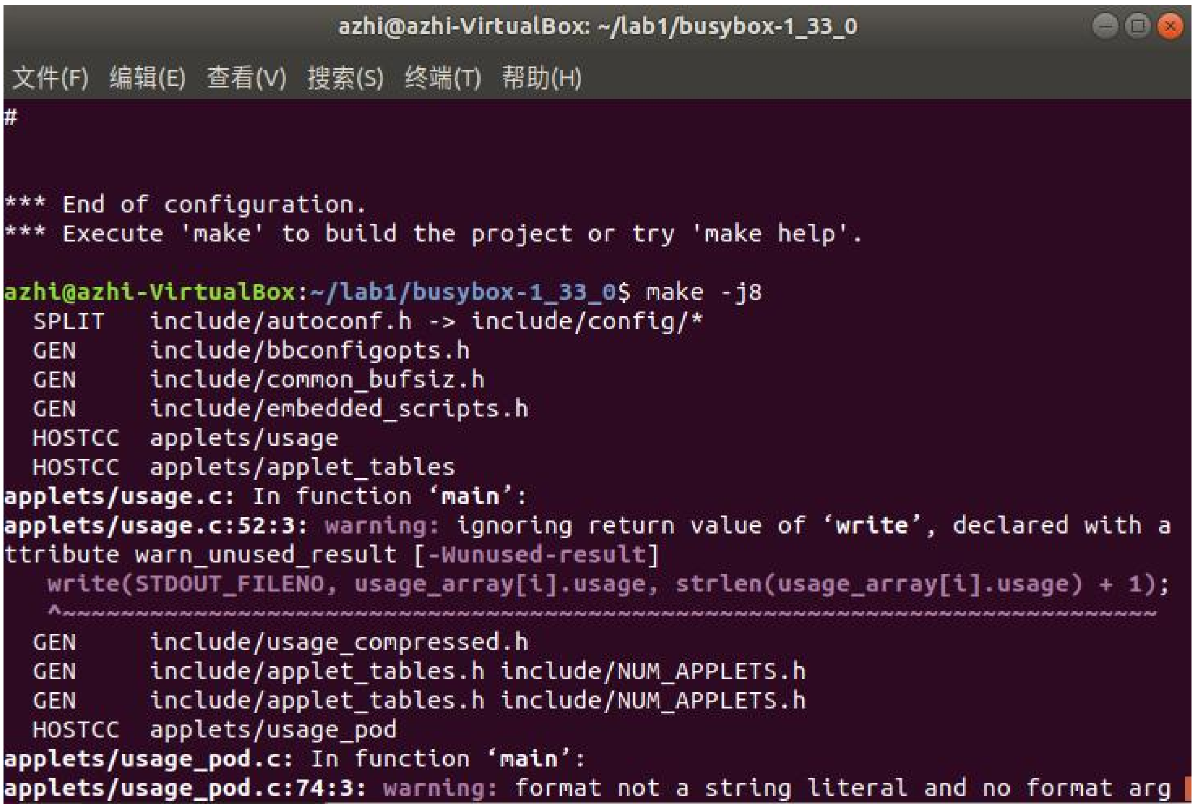
然后进入setting，在Build BusyBox as a static binary (no shared libs)处输入Y勾选，然后分别设置如下：



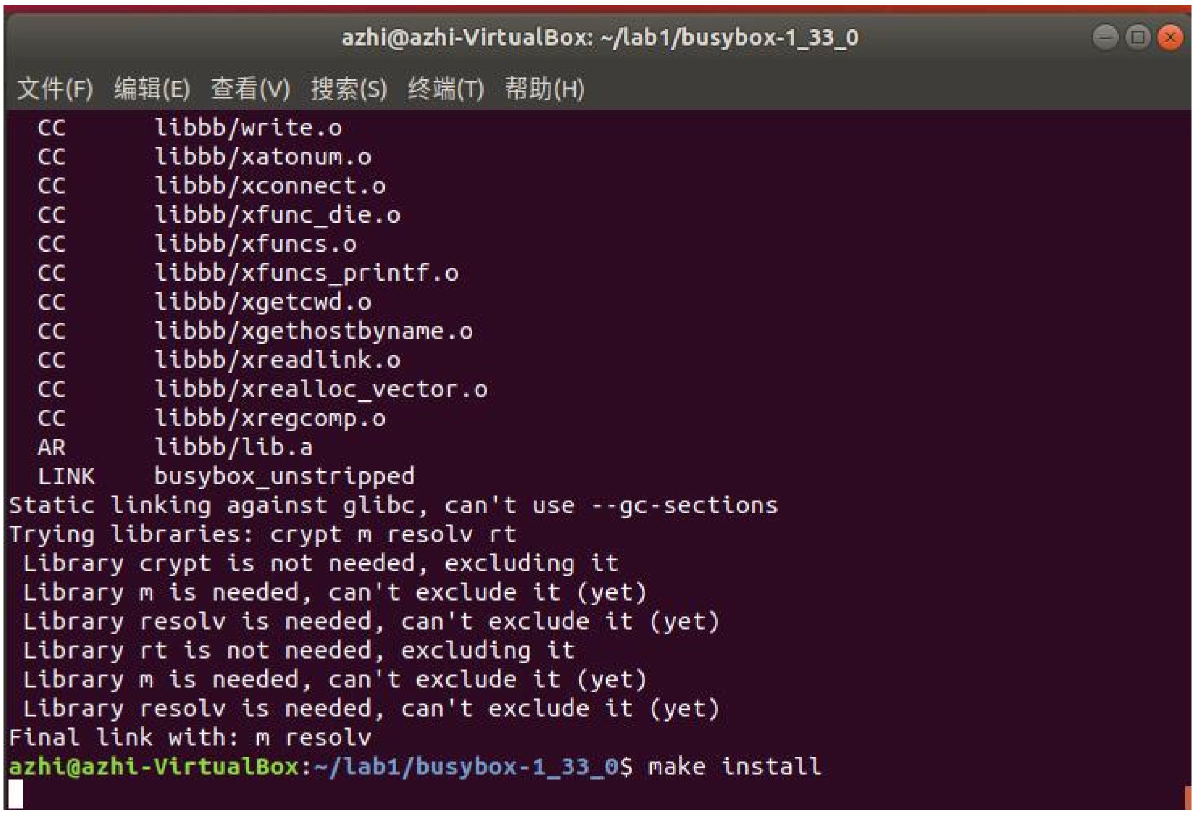
```
BusyBox 1.33.0 Configuration
                              Settings
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
  ^(-)
  [*] Build static binary (no shared libs)
  [ ] Force NOMMU build
  ()  Cross compiler prefix
  ()  Path to sysroot
  (-m32 -march=i386) Additional CFLAGS
  (-m32) Additional LDFLAGS
  ()  Additional LDLIBS
  [ ] Avoid using GCC-specific code constructs
  [*] Use -mpreferred-stack-boundary=2 on i386 arch
  [*] Use -static-libgcc
  (+)

          <Select>    < Exit >    < Help >
```

保存退出，然后编译。输入命令

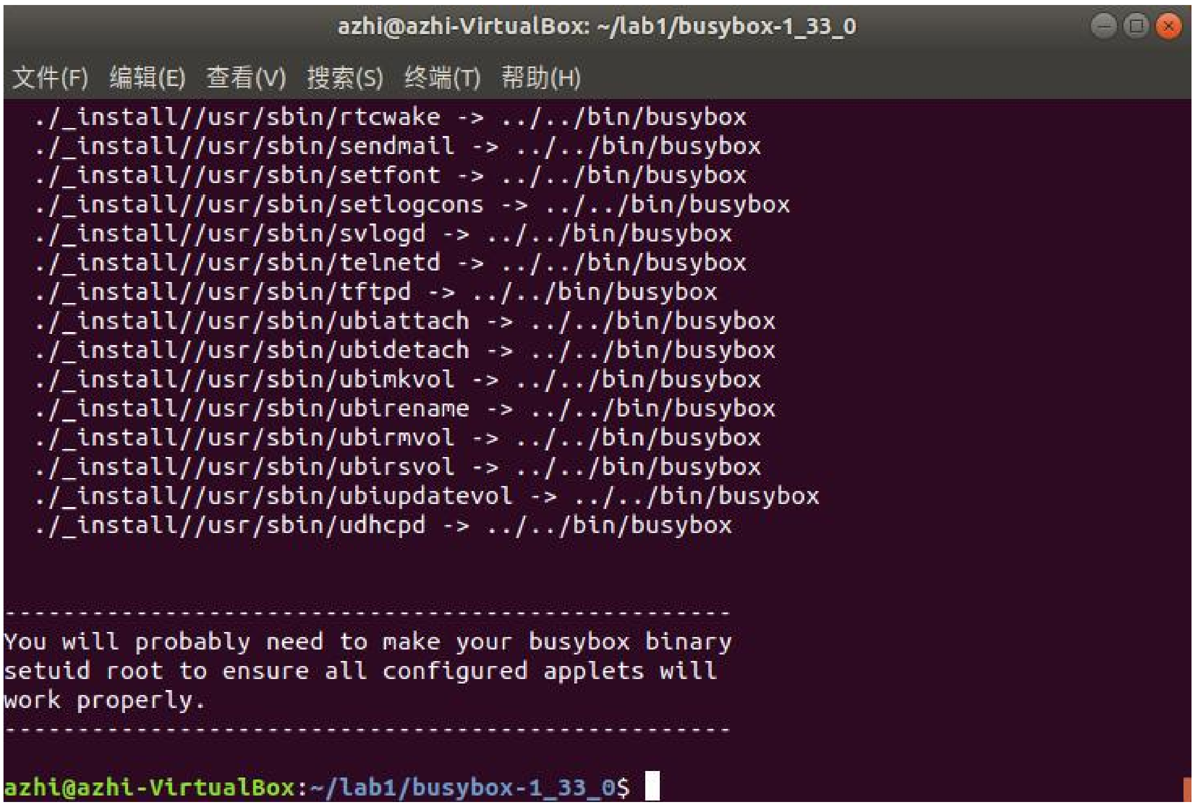
```
make -j8
make install
```


azhi@azhi-VirtualBox: ~/lab1/busybox-1_33_0
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
#


*** End of configuration.
*** Execute 'make' to build the project or try 'make help'.

azhi@azhi-VirtualBox:~/lab1/busybox-1_33_0$ make -j8
  SPLIT   include/autoconf.h -> include/config/*
  GEN     include/bbconfigopts.h
  GEN     include/common_bufsiz.h
  GEN     include/embedded_scripts.h
  HOSTCC  applets/usage
  HOSTCC  applets/applet_tables
applets/usage.c: In function 'main':
applets/usage.c:52:3: warning: ignoring return value of 'write', declared with a
ttribute warn_unused_result [-Wunused-result]
   write(STDOUT_FILENO, usage_array[i].usage, strlen(usage_array[i].usage) + 1);
   ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  GEN     include/usage_compressed.h
  GEN     include/applet_tables.h include/NUM_APPLETS.h
  GEN     include/applet_tables.h include/NUM_APPLETS.h
  HOSTCC  applets/usage_pod
applets/usage_pod.c: In function 'main':
applets/usage_pod.c:74:3: warning: format not a string literal and no format arg
```


azhi@azhi-VirtualBox: ~/lab1/busybox-1_33_0
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
  CC      libbb/write.o
  CC      libbb/xatonum.o
  CC      libbb/xconnect.o
  CC      libbb/xfunc_die.o
  CC      libbb/xfuncs.o
  CC      libbb/xfuncs_printf.o
  CC      libbb/xgetcwd.o
  CC      libbb/xgethostbyname.o
  CC      libbb/xreadlink.o
  CC      libbb/xrealloc_vector.o
  CC      libbb/xregcomp.o
  AR      libbb/lib.a
  LINK    busybox_unstripped
Static linking against glibc, can't use --gc-sections
Trying libraries: crypt m resolv rt
 Library crypt is not needed, excluding it
 Library m is needed, can't exclude it (yet)
 Library resolv is needed, can't exclude it (yet)
 Library rt is not needed, excluding it
 Library m is needed, can't exclude it (yet)
 Library resolv is needed, can't exclude it (yet)
Final link with: m resolv
azhi@azhi-VirtualBox:~/lab1/busybox-1_33_0$ make install
```

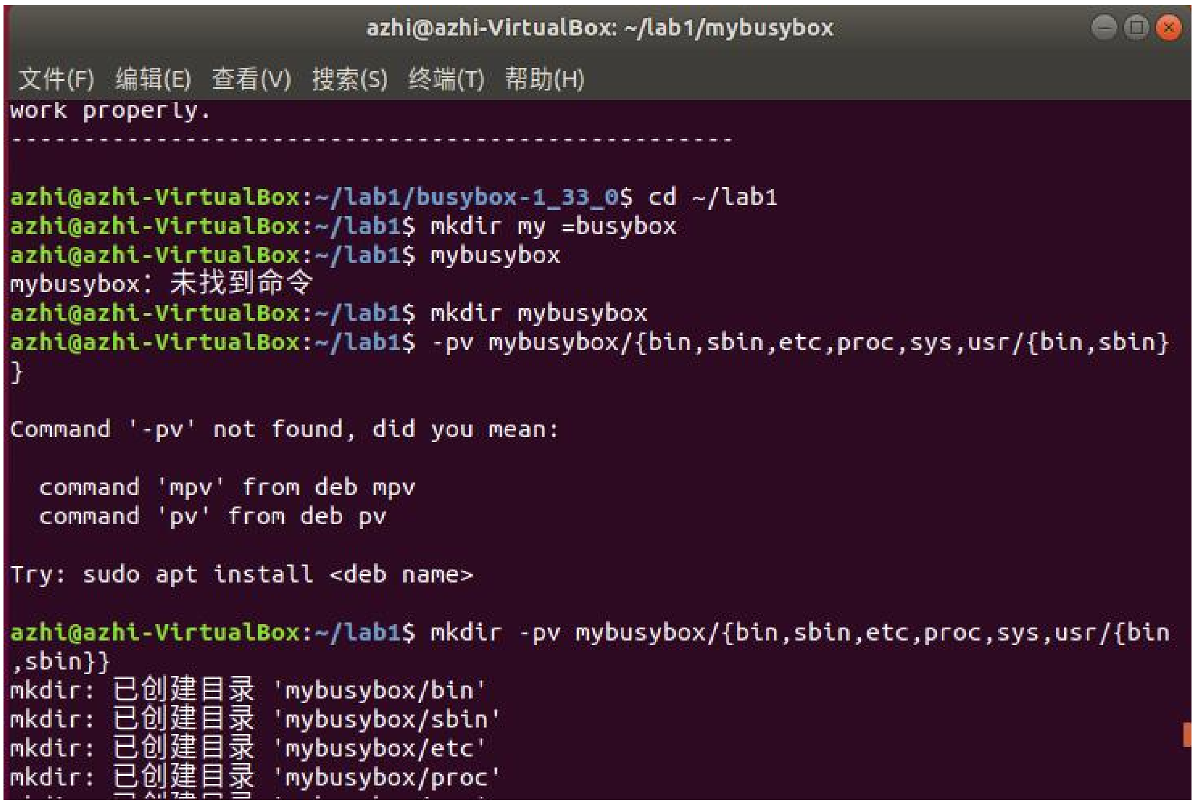

azhi@azhi-VirtualBox: ~/lab1/busybox-1_33_0
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
  ./_install//usr/sbin/rtcwake -> ../../bin/busybox
  ./_install//usr/sbin/sendmail -> ../../bin/busybox
  ./_install//usr/sbin/setfont -> ../../bin/busybox
  ./_install//usr/sbin/setlogcons -> ../../bin/busybox
  ./_install//usr/sbin/svlogd -> ../../bin/busybox
  ./_install//usr/sbin/telnetd -> ../../bin/busybox
  ./_install//usr/sbin/tftpd -> ../../bin/busybox
  ./_install//usr/sbin/ubiattach -> ../../bin/busybox
  ./_install//usr/sbin/ubidetach -> ../../bin/busybox
  ./_install//usr/sbin/ubimkvol -> ../../bin/busybox
  ./_install//usr/sbin/ubirename -> ../../bin/busybox
  ./_install//usr/sbin/ubirmvol -> ../../bin/busybox
  ./_install//usr/sbin/ubirsvol -> ../../bin/busybox
  ./_install//usr/sbin/ubiupdatevol -> ../../bin/busybox
  ./_install//usr/sbin/udhcpd -> ../../bin/busybox


--------------------------------------------------
You will probably need to make your busybox binary
setuid root to ensure all configured applets will
work properly.
--------------------------------------------------

azhi@azhi-VirtualBox:~/lab1/busybox-1_33_0$ 
```

3、制作initramfs

将安装在_install目录下的文件和目录取出放在`~/lab1/mybusybox`处。输入命令：

```
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
```


azhi@azhi-VirtualBox: ~/lab1/mybusybox
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

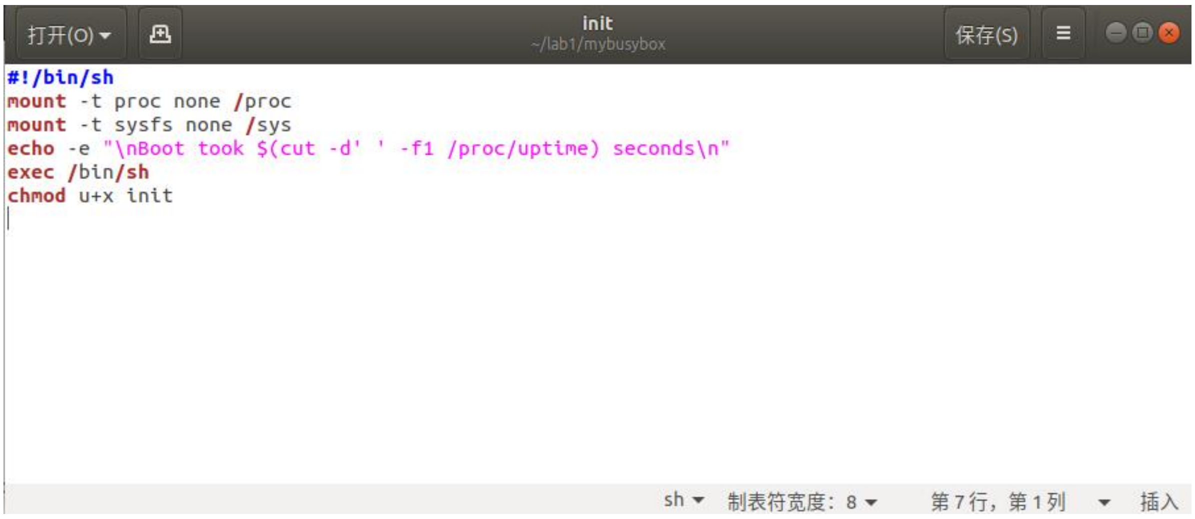
```
work properly.
--------------------------------------------------

azhi@azhi-VirtualBox:~/lab1/busybox-1_33_0$ cd ~/lab1
azhi@azhi-VirtualBox:~/lab1$ mkdir my =busybox
azhi@azhi-VirtualBox:~/lab1$ mybusybox
mybusybox: 未找到命令
azhi@azhi-VirtualBox:~/lab1$ mkdir mybusybox
azhi@azhi-VirtualBox:~/lab1$ -pv mybusybox/{bin,sbin,etc,proc,sys,usr/{bin,sbin}
}

Command '-pv' not found, did you mean:

  command 'mpv' from deb mpv
  command 'pv' from deb pv

Try: sudo apt install <deb name>

azhi@azhi-VirtualBox:~/lab1$ mkdir -pv mybusybox/{bin,sbin,etc,proc,sys,usr/{bin
,sbin}}
mkdir: 已创建目录 'mybusybox/bin'
mkdir: 已创建目录 'mybusybox/sbin'
mkdir: 已创建目录 'mybusybox/etc'
mkdir: 已创建目录 'mybusybox/proc'
```

initramfs需要一个init程序，可以写一个简单的shell脚本作为init。用`gedit`打开文件`init`，复制入如下内容，保存退出。

```
#!/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
```

加上执行权限。

`chmod u+x init`

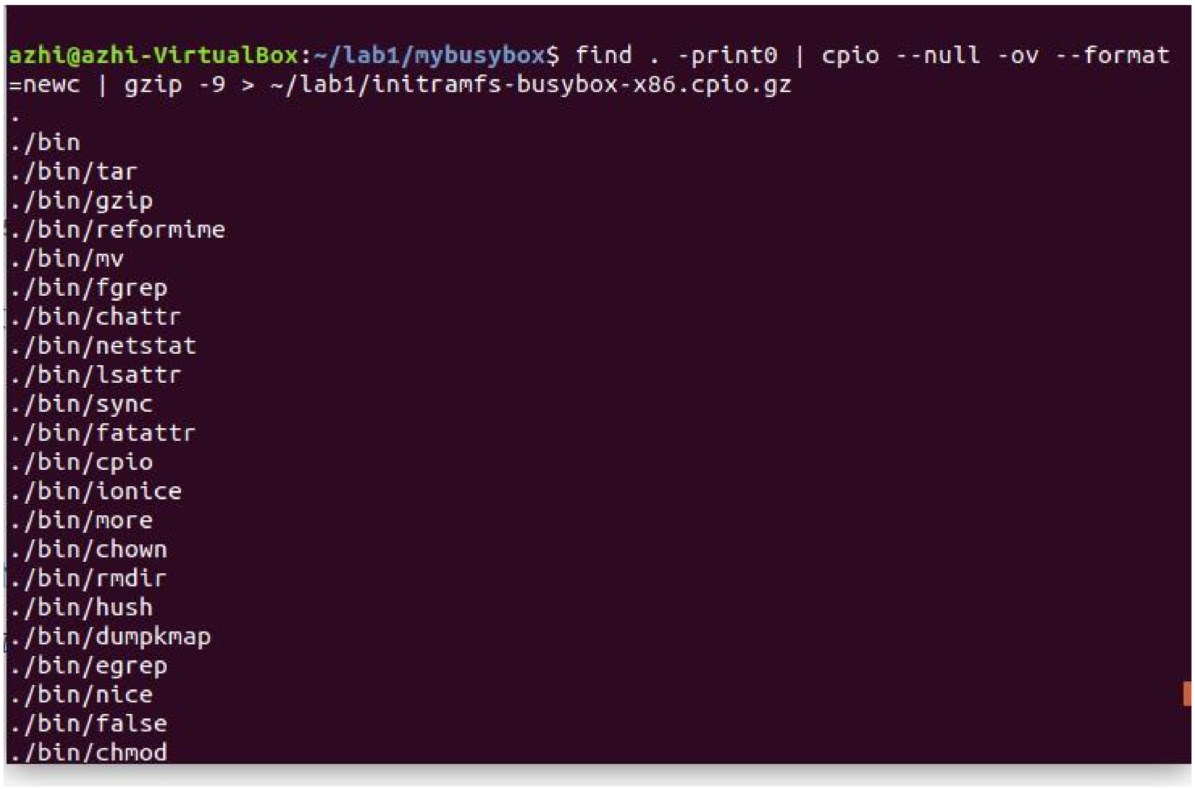


```
#!/bin/sh
mount -t proc none /proc
mount -t sysfs none /sys
echo -e "\nBoot took $(cut -d' ' -f1 /proc/uptime) seconds\n"
exec /bin/sh
chmod u+x init
```

将x86-busybox下面的内容打包归档成cpio文件，以供Linux内核做initramfs启动执行。输入

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

正在打包：



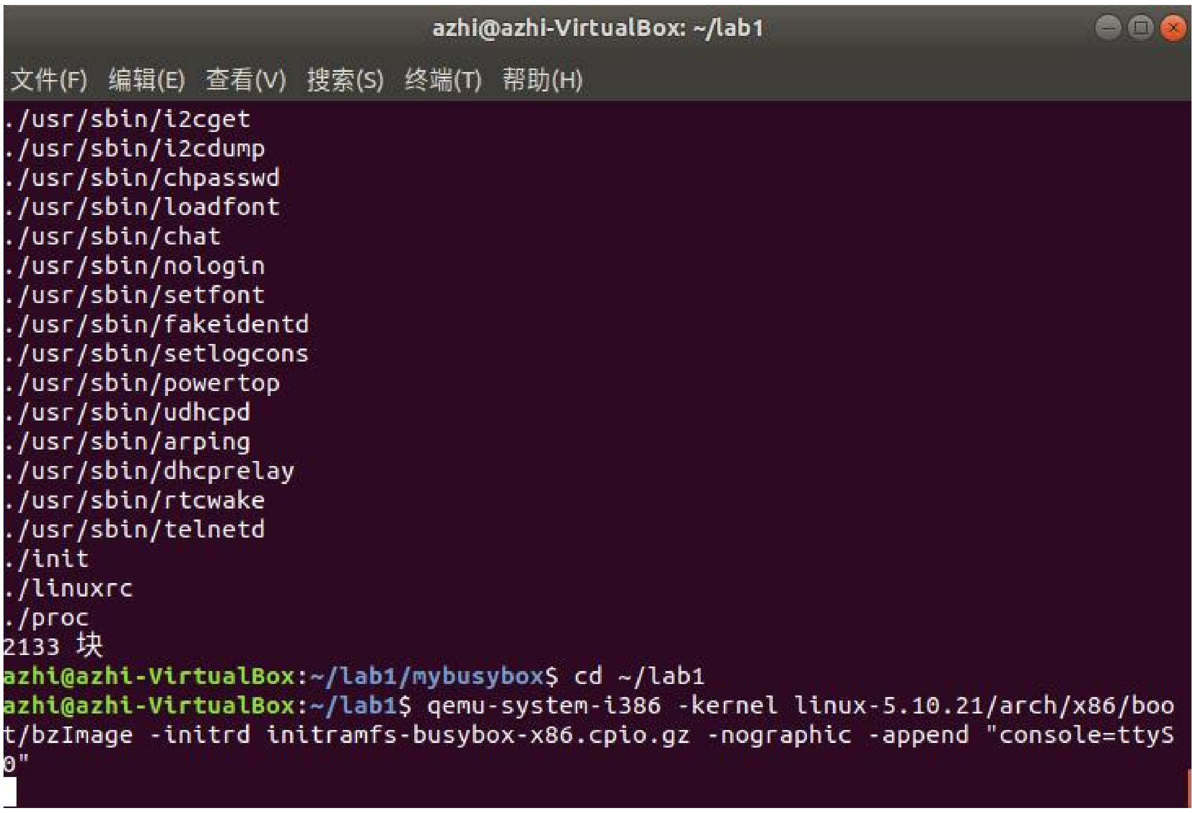
```
azhi@azhi-VirtualBox:~/lab1/mybusybox$ find . -print0 | cpio --null -ov --format
=newc | gzip -9 > ~/lab1/initramfs-busybox-x86.cpio.gz
.
./bin
./bin/tar
./bin/gzip
./bin/reformime
./bin/mv
./bin/fgrep
./bin/chattr
./bin/netstat
./bin/lsattr
./bin/sync
./bin/fatattr
./bin/cpio
./bin/ionice
./bin/more
./bin/chown
./bin/rmdir
./bin/hush
./bin/dumpkmap
./bin/egrep
./bin/nice
./bin/false
./bin/chmod
```

4、加载busybox

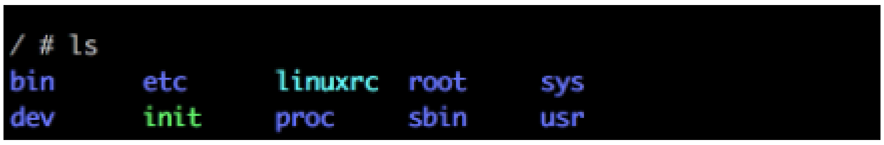
输入命令：

```
cd ~/lab1
qemu-system-i386 -kernel linux-5.10.19/arch/x86/boot/bzImage -initrd initramfs-busybox-x86.cpio.gz -nographic -append "console=ttyS0"
```

然后使用`ls`命令即可看到当前文件夹。


azhi@azhi-VirtualBox: ~/lab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
./usr/sbin/i2cget
./usr/sbin/i2cdump
./usr/sbin/chpasswd
./usr/sbin/loadfont
./usr/sbin/chat
./usr/sbin/nologin
./usr/sbin/setfont
./usr/sbin/fakeidentd
./usr/sbin/setlogcons
./usr/sbin/powertop
./usr/sbin/udhcpd
./usr/sbin/arping
./usr/sbin/dhcprelay
./usr/sbin/rtcwake
./usr/sbin/telnetd
./init
./linuxrc
./proc
2133 块
azhi@azhi-VirtualBox:~/lab1/mybusybox$ cd ~/lab1
azhi@azhi-VirtualBox:~/lab1$ qemu-system-i386 -kernel linux-5.10.21/arch/x86/boo
t/bzImage -initrd initramfs-busybox-x86.cpio.gz -nographic -append "console=ttyS
0"
```



```
/ # ls
bin       etc       linuxrc   root      sys
dev       init      proc      sbin      usr
```

六、实验感想：

本次实验主要是对Linux环境的搭建，做好本次实验将是日后做实验的基础。