

操作系统原理 实验一

个人信息

【院系】 计算机学院

【专业】 计算机科学与技术

【学号】 20337263

【姓名】 俞泽斌

实验题目

编译内核/利用已有内核构建OS

实验目的

1. 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。
2. 利用精简的Busybox工具集构建简单的OS，熟悉现代操作系统的构建过程。
3. 熟悉编译环境、相关工具集，并能够实现内核远程调试。

实验要求

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。

实验方案

一开始的操作就是下载一个VMware软件，然后在其中配置了一个Ubuntu的虚拟机，进行一系列常规的分辨率和语言改动之后进入正题

首先是对于下载速度的升级了，换成了aliyun的镜像来提高下载速度，也就是在/etc/apt/sources.list里加入了阿里云的一些链接，也是从网上复制过来的，然后按照tutorial里的操作步骤，进行c++环境的配置，并且通过命令行的各种操作，安装上各种必要软件比如qemu，nasm等等，Ubuntu20.04版本的

nasm似乎是2.14的，与tutorial里的nasm2.15有所出入，但是在后续实验中好像没有大的影响。

接下来就是内核的下载和配置，首先从Linux下载内核到lab1文件夹，然后经过解压等一系列操作，将内核编译成i386 32位版本并编译内核，检查Linux压缩镜像linux-5.10.19/arch/x86/boot/bzImage和符号表linux-5.10.19/vmlinux是否已经生成

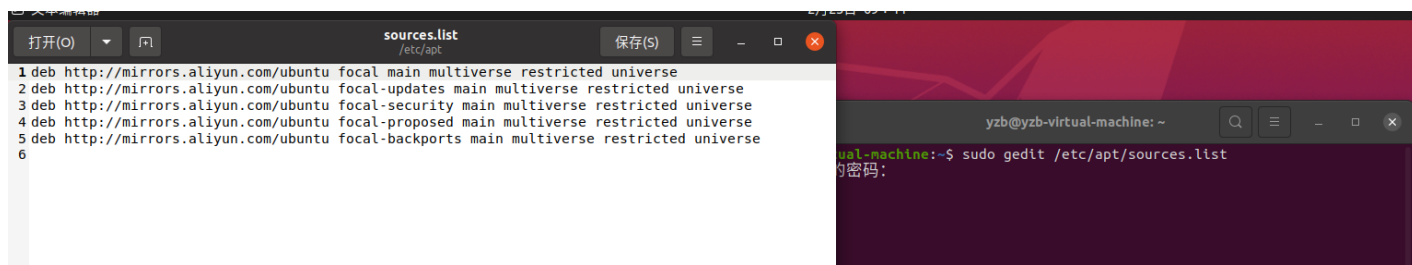
然后是gdb调试的步骤:使用qemu启动内核并开启远程调试。在另外一个Terminal下启动gdb，并在gdb下，加载符号表，在gdb下，连接已经启动的qemu进行调试，再通过设置断点并运行。

之后进行制作Initramfs，先在lab1下写一个简单的helloworld文件，然后将文件保存在~/lab1/helloworld.c中，然后将上面代码编译成32位可执行文件。用cpio打包initramfs。启动内核，并加载initramfs。

最后是编译并启动Busybox，从Busybox官网下载Busybox到/lab1，然后解压。这里采用的是busybox1.33.2的版本，也与tutorial里的有所出入，但是影响不大，接着按照tutorial进行一些设置，并进行编译，最后制作Initramfs将安装在_install目录下的文件和目录取出放在~/lab1/mybusybox处。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。加上执行权限并将x86-busybox内容打包归档成cpio文件，以供Linux内核做initramfs启动执行。最后加载busybox，即可用ls命令。

包括：硬件或虚拟机配置方法、软件工具与作用、方案的思想、相关原理、程序流程、算法和数据结构、程序关键模块，结合代码与程序中的位置位置进行解释。不得抄袭，否则按作弊处理。

实验过程



这里就是有关于源的改变了，我采用的是阿里云的镜像来提高下载速度。其实一开始想用的是清华源，也是老师在教程里面所推荐的，但是换源的时候出现了找不到release文件的报错，当时上网查就直接换成

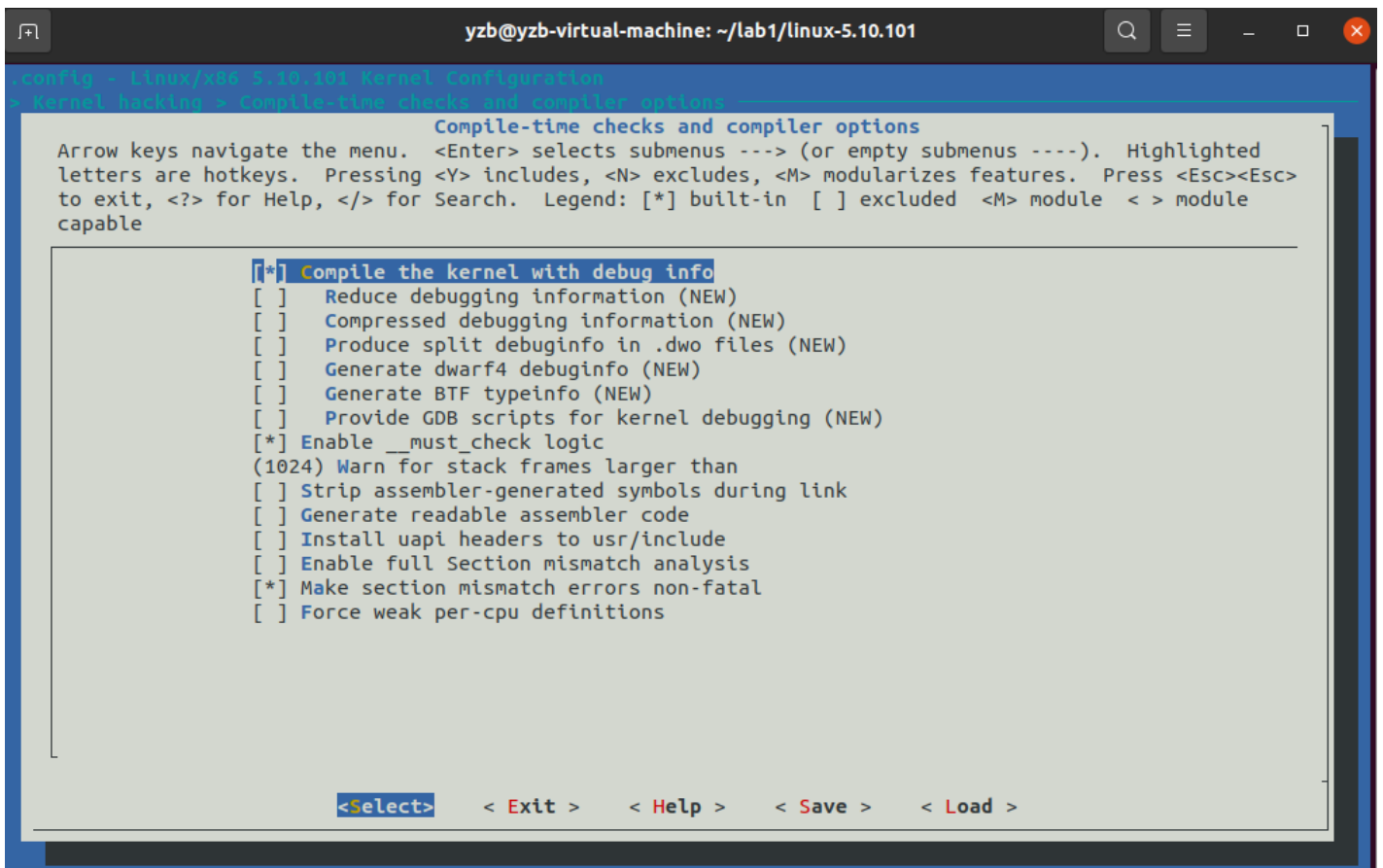
阿里云了，后来在群里的讨论下才发现就是http和https的区别

```
yzb@yzb-virtual-machine: ~  
获取:13 http://mirrors.aliyun.com/ubuntu focal-updates/universe amd64 Packages [905 kB]  
获取:14 http://mirrors.aliyun.com/ubuntu focal-updates/universe i386 Packages [669 kB]  
获取:15 http://mirrors.aliyun.com/ubuntu focal-updates/universe Translation-en [201 kB]  
获取:16 http://mirrors.aliyun.com/ubuntu focal-updates/universe amd64 DEP-11 Metadata [390 kB]  
获取:17 http://mirrors.aliyun.com/ubuntu focal-security/main amd64 Packages [1,265 kB]  
获取:18 http://mirrors.aliyun.com/ubuntu focal-security/main i386 Packages [383 kB]  
获取:19 http://mirrors.aliyun.com/ubuntu focal-security/main Translation-en [221 kB]  
获取:20 http://mirrors.aliyun.com/ubuntu focal-security/main amd64 DEP-11 Metadata [40.7 kB]  
获取:21 http://mirrors.aliyun.com/ubuntu focal-security/main amd64 c-n-f Metadata [9,624 B]  
获取:22 http://mirrors.aliyun.com/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [2,464 B]  
获取:23 http://mirrors.aliyun.com/ubuntu focal-security/universe amd64 Packages [679 kB]  
获取:24 http://mirrors.aliyun.com/ubuntu focal-security/universe i386 Packages [535 kB]  
获取:25 http://mirrors.aliyun.com/ubuntu focal-security/universe Translation-en [116 kB]  
获取:26 http://mirrors.aliyun.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [66.4 kB]  
获取:27 http://mirrors.aliyun.com/ubuntu focal-proposed/main amd64 DEP-11 Metadata [3,896 B]  
获取:28 http://mirrors.aliyun.com/ubuntu focal-proposed/multiverse amd64 DEP-11 Metadata [940 B]  
获取:29 http://mirrors.aliyun.com/ubuntu focal-proposed/universe i386 Packages [161 kB]  
获取:30 http://mirrors.aliyun.com/ubuntu focal-proposed/universe amd64 Packages [213 kB]  
获取:31 http://mirrors.aliyun.com/ubuntu focal-proposed/universe Translation-en [38.6 kB]  
获取:32 http://mirrors.aliyun.com/ubuntu focal-proposed/universe amd64 DEP-11 Metadata [25.2 kB]  
获取:33 http://mirrors.aliyun.com/ubuntu focal-backports/main amd64 DEP-11 Metadata [7,996 B]  
获取:34 http://mirrors.aliyun.com/ubuntu focal-backports/universe amd64 Packages [22.0 kB]  
获取:35 http://mirrors.aliyun.com/ubuntu focal-backports/universe i386 Packages [12.1 kB]  
获取:36 http://mirrors.aliyun.com/ubuntu focal-backports/universe Translation-en [15.2 kB]  
获取:37 http://mirrors.aliyun.com/ubuntu focal-backports/universe amd64 DEP-11 Metadata [30.5 kB]  
获取:38 http://mirrors.aliyun.com/ubuntu focal-backports/universe DEP-11 48x48 Icons [13.2 kB]  
获取:39 http://mirrors.aliyun.com/ubuntu focal-backports/universe DEP-11 64x64 Icons [21.6 kB]  
获取:40 http://mirrors.aliyun.com/ubuntu focal-backports/universe amd64 c-n-f Metadata [728 B]  
已下载 9,463 kB, 耗时 13秒 (723 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  
有 395 个软件包可以升级。请执行 'apt list --upgradable' 来查看它们。  
yzb@yzb-virtual-machine:~$
```

这是sudo apt update命令后的运行截图，可以看到源换成功了，也输出了一些过程

```
有 395 个软件包可以升级。请执行 'apt list --upgradable' 来查看它们。  
yzb@yzb-virtual-machine:~$ gcc -v  
Using built-in specs.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/9/lto-wrapper  
OFFLOAD_TARGET_NAMES=nvptx-none:hsa  
OFFLOAD_TARGET_DEFAULT=1  
Target: x86_64-linux-gnu  
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 9.4.0-1ubuntu1~20.04' --with-bugurl=file:///usr/share/doc/gcc-9/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,gm2 --prefix=/usr --with-gc-c-major-version-only --program-suffix=-9 --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-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib=auto --enable-objc-gc=auto --enable-multilarch --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=/build/gcc-9-yTrUTS/gcc-9-9.4.0/debian/tmp-nvptx/usr,hsa --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 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04)  
yzb@yzb-virtual-machine:~$
```

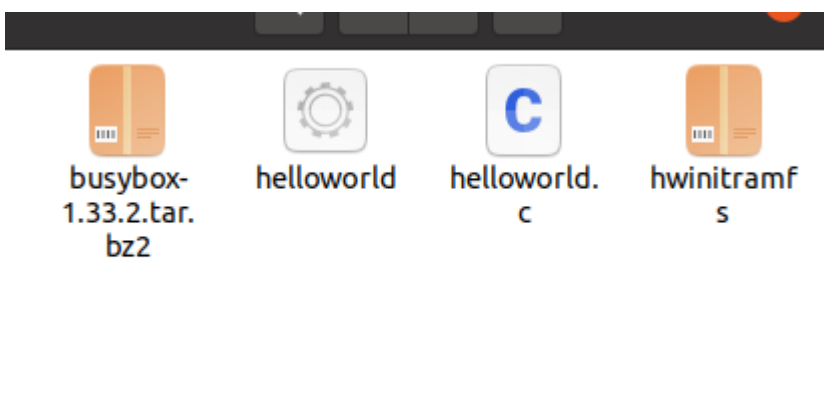
gcc-v的命令，可以看到最后输出了gcc的版本号 gcc version 9.4.0,也就是表明gcc已经安装成功，c语言环境已经配置完成。



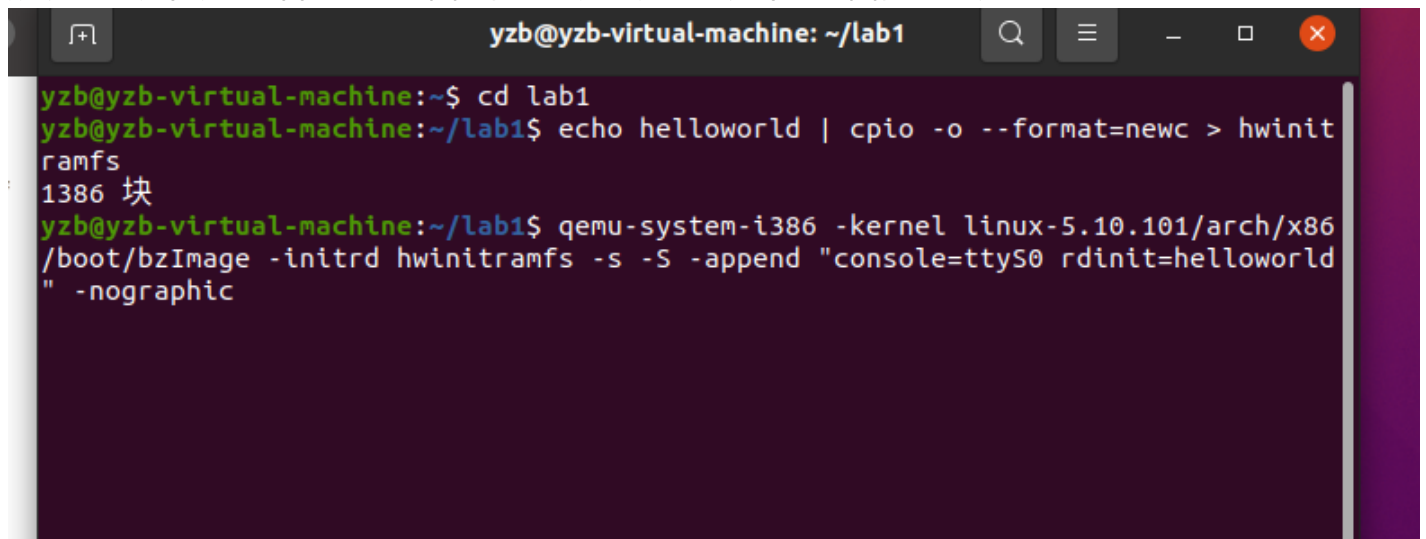
这里将内核编译成i386 32位版本，也就是tutorial中所提到的选项操作，选完后保存退出



helloworld的c代码，有关gdb调试的那部分，因为后文中有直接使用代码进行调试的，比开始这里的直接调试并未输出结果的更有效果，所以这里不加赘述

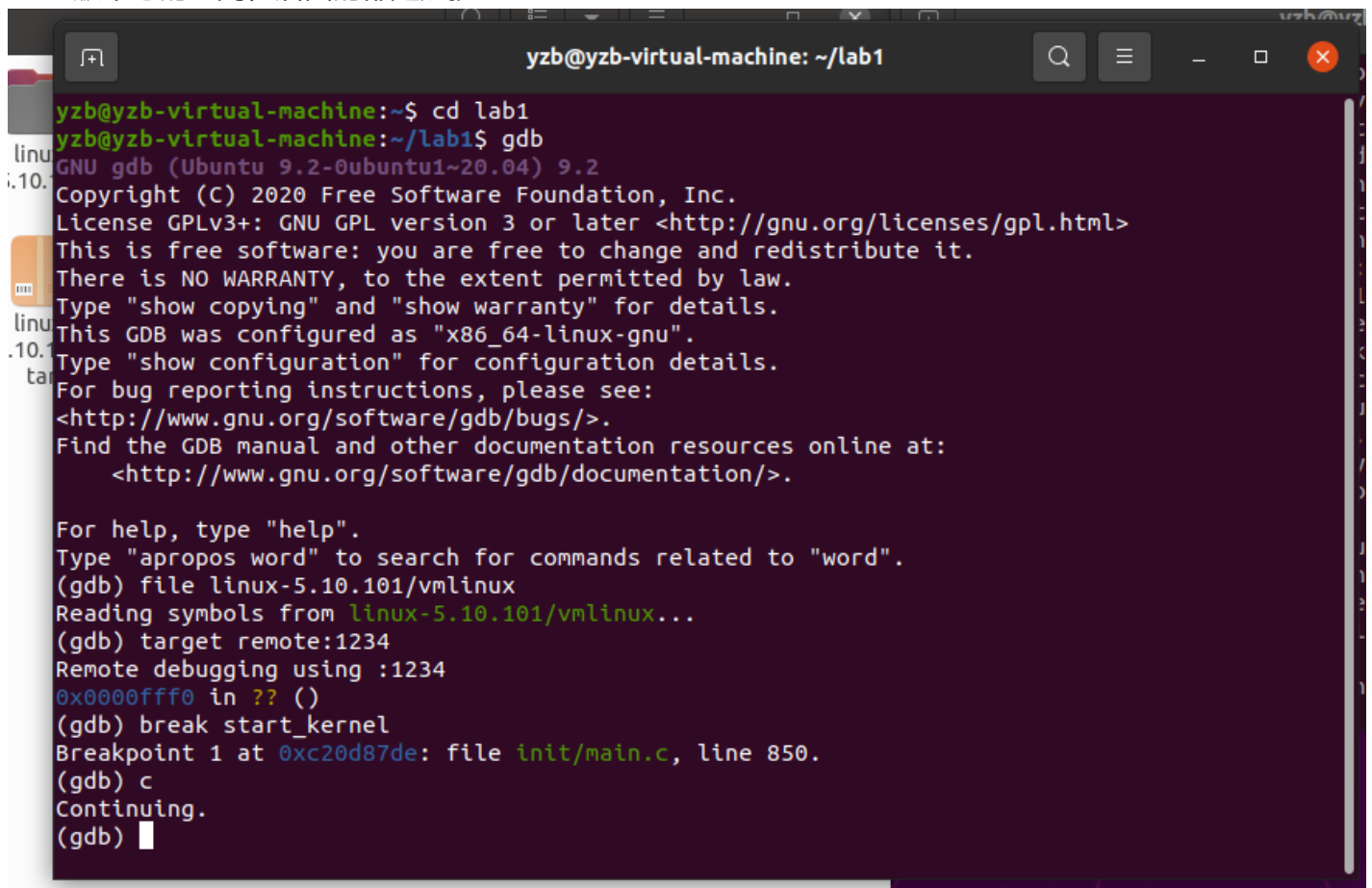


可以看到helloworld.c文件已经被编译成了可以执行的文件，因为一开始第一次进行编译的时候没有截图，所以这里就不放控制台终端的截图了，转而用的是文件夹中的截图，也可以看出里面存在了编译后的文件



```
yzb@yzb-virtual-machine: ~/lab1
yzb@yzb-virtual-machine:~$ cd lab1
yzb@yzb-virtual-machine:~/lab1$ echo helloworld | cpio -o --format=newc > hwinitramfs
1386 块
yzb@yzb-virtual-machine:~/lab1$ qemu-system-i386 -kernel linux-5.10.101/arch/x86/boot/bzImage -initrd hwinitramfs -s -S -append "console=ttyS0 rdinit=helloworld" -nographic
```

这里就采用了用cpio打包initramfs和启动内核，并加载initramfs的操作步骤，其实主要注意的还是有关于linux版本号的差别，后面的都是照抄



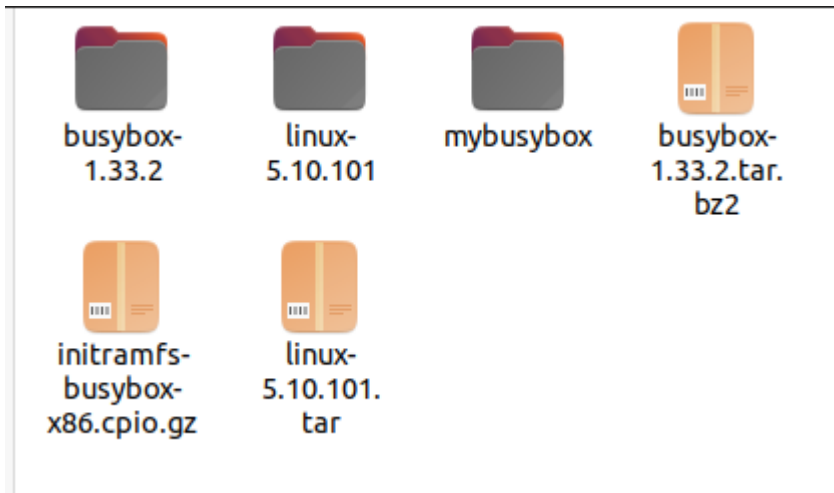
```
yzb@yzb-virtual-machine:~$ cd lab1
yzb@yzb-virtual-machine:~/lab1$ gdb
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute 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-5.10.101/vmlinux
Reading symbols from linux-5.10.101/vmlinux...
(gdb) target remote:1234
Remote debugging using :1234
0x00000000 in ?? ()
(gdb) break start_kernel
Breakpoint 1 at 0xc20d87de: file init/main.c, line 850.
(gdb) c
Continuing.
(gdb)
```

这是gdb调试界面，也是按照tutorial里的直接进行操作，后面可以看到在另一个终端页面，也就是initramfs的加载页面出现了helloworld的界面


```
yzb@yzb-virtual-machine: ~/lab1
[ 2.837805] NET: Registered protocol family 17
[ 2.844292] Key type dns_resolver registered
[ 2.845544] mce: Unable to init MCE device (rc: -5)
[ 2.849543] IPI shorthand broadcast: enabled
[ 2.849956] sched_clock: Marking stable (2773670849, 75495005)->(3023098412,)
[ 2.854544] registered taskstats version 1
[ 2.855082] Loading compiled-in X.509 certificates
[ 2.858944] PM: Magic number: 10:284:509
[ 2.859438] ata_piix 0000:00:01.1: hash matches
[ 2.860003] printk: console [netcon0] enabled
[ 2.860381] netconsole: network logging started
[ 2.869317] cfg80211: Loading compiled-in X.509 certificates for regulatory e
[ 2.888218] kworker/u2:0 (59) used greatest stack depth: 7020 bytes left
[ 2.901086] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[ 2.903647] platform regulatory.0: Direct firmware load for regulatory.db fa2
[ 2.904245] cfg80211: failed to load regulatory.db
[ 2.904904] ALSA device list:
[ 2.904989]   No soundcards found.
[ 3.008884] Freeing unused kernel image (initmem) memory: 676K
[ 3.014775] Write protecting kernel text and read-only data: 15236k
[ 3.015316] Run helloworld as init process
lab1: Hello World
[ 3.420009] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i8043
```

这里就第一个任务完成了



加载busybox的有关开始的一些步骤，因为第一次做的时候没有截图，就放上来一个选择文件夹里的截图，可以看到busybox的安装以及制作Initramfs并把它命名为mybusybox的生成都是成功了的

```
init
~/lab1/mybusybox
1 #!/bin/sh
2 mount -t proc none /proc
3 mount -t sysfs none /sys
4 echo -e "\nBoot took $(cut -d' ' -f1 /proc/uptime) seconds\n"
5 exec /bin/sh

yzb@yzb-virtual-machine: ~$ cd lab1
yzb@yzb-virtual-machine: ~/lab1$ cd mybusybox
yzb@yzb-virtual-machine: ~/lab1/mybusybox$ gedit init
```

shell脚本的主要内容，第一行一开始没有加上就导致后面一直出现cannot find the dictionary的情况（虽然出现了那种情况之后只要ls命令输入得够快也可以出现文件夹）但是加上这一行之后就不再一直刷屏

cannot find the dictionaryl。也可以正常地输入ls命令显示文件夹

```
yzb@yzb-virtual-machine: ~/lab1/mybusybox
./usr/sbin/setfont
./usr/sbin/tftpd
./usr/sbin/chat
./usr/sbin/i2cget
./usr/sbin/loadfont
./usr/sbin/telnetd
./usr/sbin/fakeidentd
./usr/sbin/powertop
./usr/sbin/ftpd
./usr/sbin/nbd-client
./usr/sbin/remove-shell
./usr/sbin/i2cdump
./usr/sbin/httpd
./usr/sbin/delgroup
./usr/sbin/chpasswd
./usr/sbin/ubimkvol
./usr/sbin/adduser
./usr/sbin/i2ctransfer
./usr/sbin/ubirename
./sys
./proc
./etc
4455 块
yzb@yzb-virtual-machine:~/lab1/mybusybox$
```

这是将x86-busybox下面的内容打包归档成cpio文件，以供Linux内核做initramfs启动执行后的终端界面，也可以看到是成功的

```
yzb@yzb-virtual-machine: ~/lab1
[ 2.092816] cfg80211: Loading compiled-in X.509 certificates for regulatory e
[ 2.132704] modprobe (59) used greatest stack depth: 7036 bytes left
[ 2.148266] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[ 2.150716] platform regulatory.0: Direct firmware load for regulatory.db fa2
[ 2.151492] cfg80211: failed to load regulatory.db
[ 2.152348] ALSA device list:
[ 2.152528]   No soundcards found.
[ 2.231824] Freeing unused kernel image (initmem) memory: 676K
[ 2.236327] Write protecting kernel text and read-only data: 15236k
[ 2.236967] Run /init as init process
[ 2.325452] mount (63) used greatest stack depth: 6980 bytes left
[ 2.333999] tsc: Refined TSC clocksource calibration: 2304.058 MHz
[ 2.335128] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x21362ebs
[ 2.337710] clocksource: Switched to clocksource tsc
[ 2.359500] cut (65) used greatest stack depth: 6908 bytes left

Boot took 2.28 seconds

/bin/sh: can't access tty; job control turned off
/ # [ 2.693133] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/3
ls
bin      etc      linuxrc  root     sys
dev      init     proc     sbin     usr
/ #
```

最后的操作，也就是加载busybox，加载了之后再输入ls命令，可以看到输出了其中所有的文件夹，说明busybox加载成功

包括：主要工具安装使用过程及截图结果、程序过程中的操作步骤、测试数据、输入及输出说明、遇到的问题及解决情况、关键功能或操作的截图结果。不得抄袭，否则按作弊处理。

实验总结

因为大部分的实验操作都是按照tutorial里面写的步骤来做的，所以可能理解上来说并没有特别的突出，第一次遇到的问题就是清华源的换源问题，其实上次不知道什么时候好像是给我的虚拟机换源过的，但是还是忘记掉了，一开始跟着tutorial里的步骤去做，发现存在报错是没有release文件，当时也不知道这报错是什么意思，上网查到可以换成阿里云，也就直接复制了，并未深究，后来看群里的发言才知道https应该改成http，这是其一。

然后是关于选项界面的操作，一开始还不太会用，后来慢慢理解和发现了这些选项的使用方式。对于内核的下载和编译来说，一开始也就是直接把tutorial里的代码复制过来，后来明白是把linux内核编译成i386的形式，来通过指令对一些程序进行操作。其实还有个比较小的方面是关于文件的解压，linux里的解压方式就是将文件的后缀名一个一个提到前面来进行解压，也是一件比较有趣的事情，主要就是linux内核的解压和后面busybox的解压两个方面了。同时我还学习了怎么在命令行界面下采用gdb调试，也就是在两个终端下进行编译和运行操作，也是一个比较新奇的体验

制作initramfs的方面，我也熟悉了基本的使用命令行以及 linux内核来调试和运行的步骤在lab1下写一个简单的helloworld文件，文件保存在helloworld.c中，然后将上面代码编译成32位可执行文件。用cpio打包initramfs。启动内核，并加载initramfs。同时我还学习了怎么在命令行界面下采用gdb调试，也就是在两个终端下进行编译和运行操作，也是一个比较新奇的体验

最后还有一个方面是关于shell文件的，一开始进行是按照tutorial里的文件直接复制的，后来在编译之后将要输入ls命令的前一步的时候发现一直刷屏出现cannot find the dictionary的情况，后来在查找网上资料以及看群里助教回复的方法下才改掉了shell文件，并且成功实现了最后的ls命令查看所有文件夹。

每人必需写一段，文字不少于500字，可以写心得体会、问题讨论与思考、新的设想、感言总结或提出建议等等。不得抄袭，否则按作弊处理。

参考文献

~/tutorial

[https://blog.csdn.net/m0_52672980/article/details/122315063?](https://blog.csdn.net/m0_52672980/article/details/122315063?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164584290216781685366185%2522%252C%2522scm%2522%253A%25220140713.130102334..%2522%257D&request_id=164584290216781685366185&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-122315063.pc_search_insert_es_download&utm_term=ubuntu20.04%E6%B2%A1%E6%9C%89release%E6%96%87%E4%BB%B6&spm=1018.2226.3001.4187)

[ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164584290216781685366185%2522%252C%2522scm%2522%253A%25220140713.130102334..%2522%257D&request_id=164584290216781685366185&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-122315063.pc_search_insert_es_download&utm_term=ubuntu20.04%E6%B2%A1%E6%9C%89release%E6%96%87%E4%BB%B6&spm=1018.2226.3001.4187](https://blog.csdn.net/m0_52672980/article/details/122315063?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164584290216781685366185%2522%252C%2522scm%2522%253A%25220140713.130102334..%2522%257D&request_id=164584290216781685366185&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-122315063.pc_search_insert_es_download&utm_term=ubuntu20.04%E6%B2%A1%E6%9C%89release%E6%96%87%E4%BB%B6&spm=1018.2226.3001.4187)