# RISC-V QEMU 上运行 Linux 系统

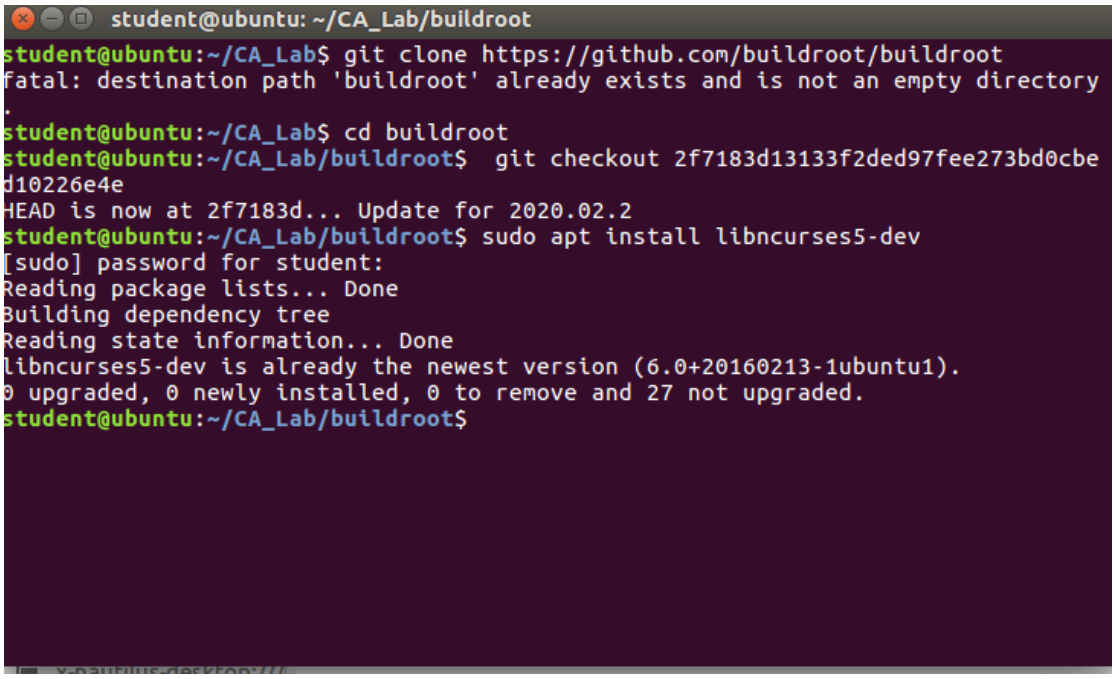
1、实验目的

通过在 RISC-V QEMU 上运行 Linux 操作系统，进一步掌握 QEMU 系统模式的使用方法、了解交叉工具链的作用、Buildroot 的主要功能和 Linux 系统的基本原理。

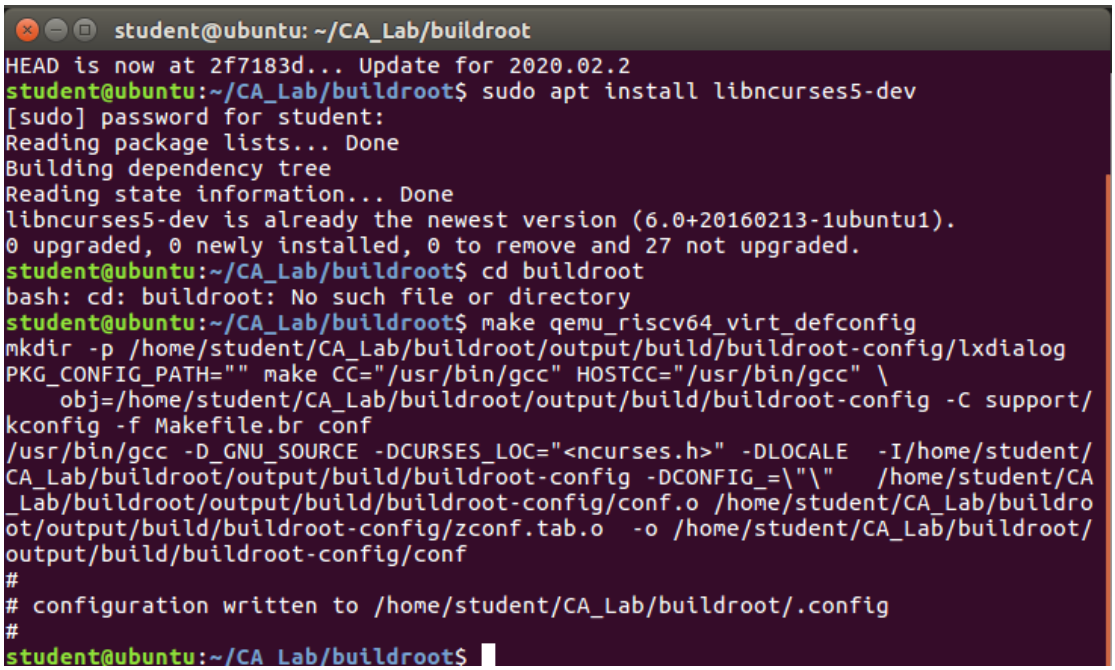
2、实验步骤（包括实验结果，数据记录、截图等）

（1）使用 Buildroot 搭建 Linux 系统的配置和编译过程

1. 用 git 下载 Buildroot，注意下载完仓库后要切换分支到指定版本，切换分支后直接用 apt 下载

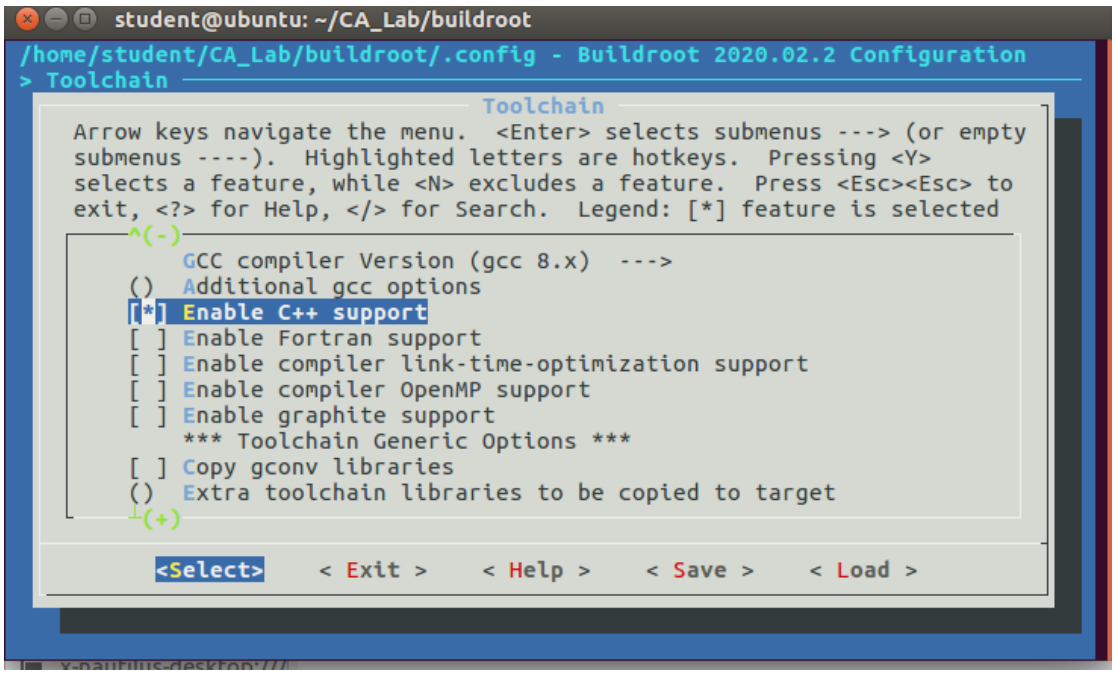



2. 输入 make qemu_riscv64_virt_defconfig 指令，导入针对 RISCV64 架构的默认配置文件：

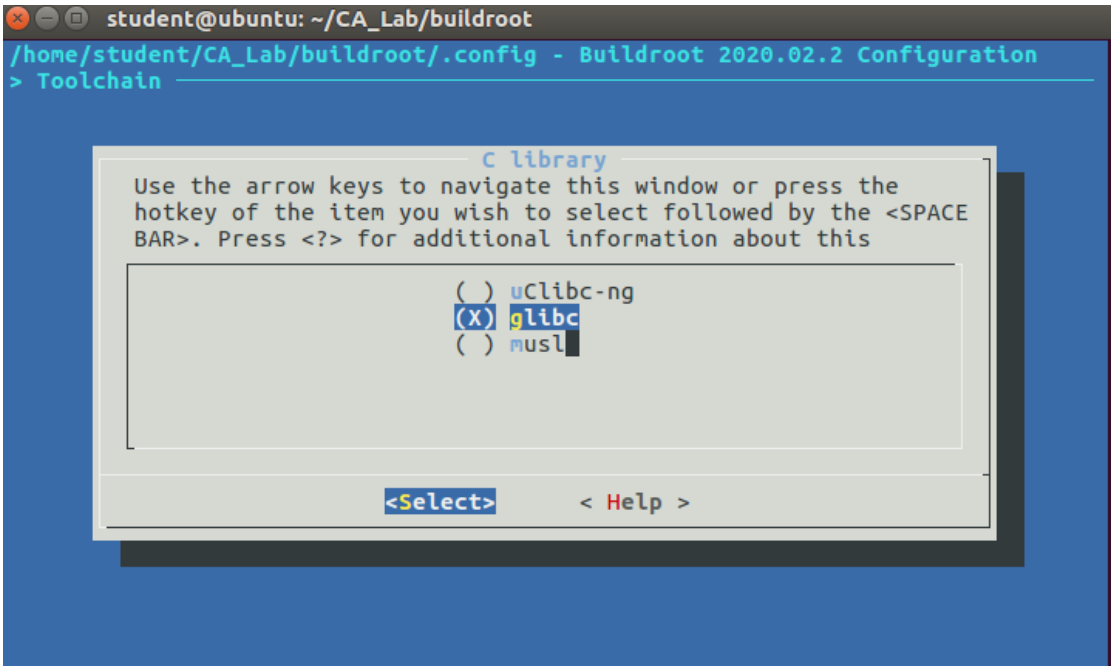



3. 在默认配置的基础上进行修改配置：

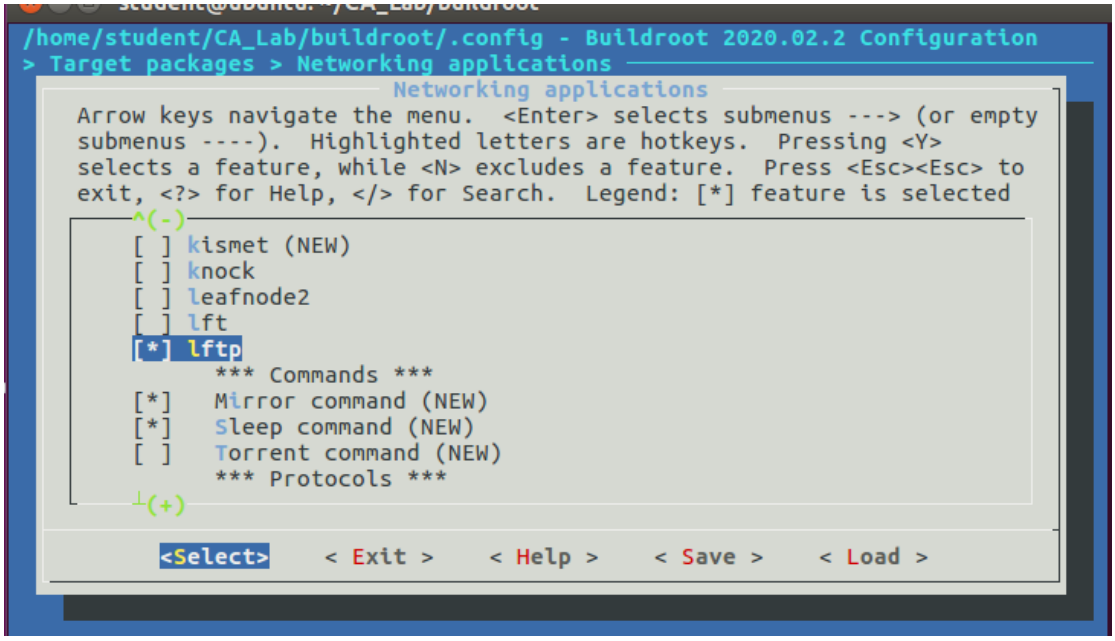
1. 在 toolchain 目录下勾选 Enable c++ support 选项



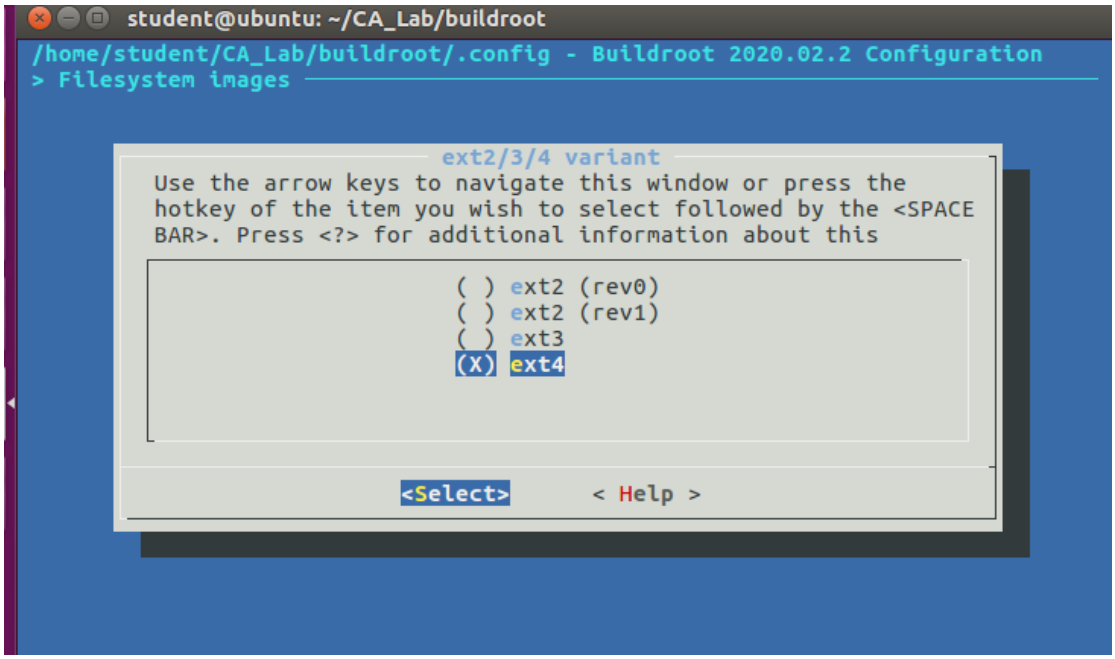

2. 在 toolchain 目录下查看 C library 查看 GNU 的标准 glibc 作为 C 库

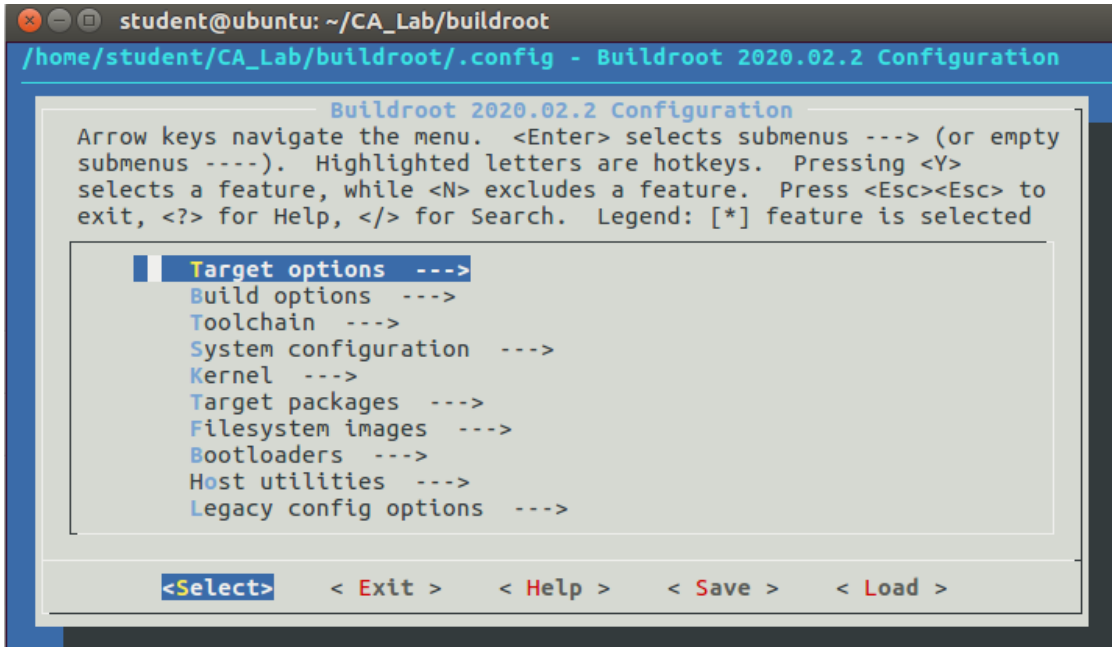



3. 在 Networking apllications 目录下勾选 lftp

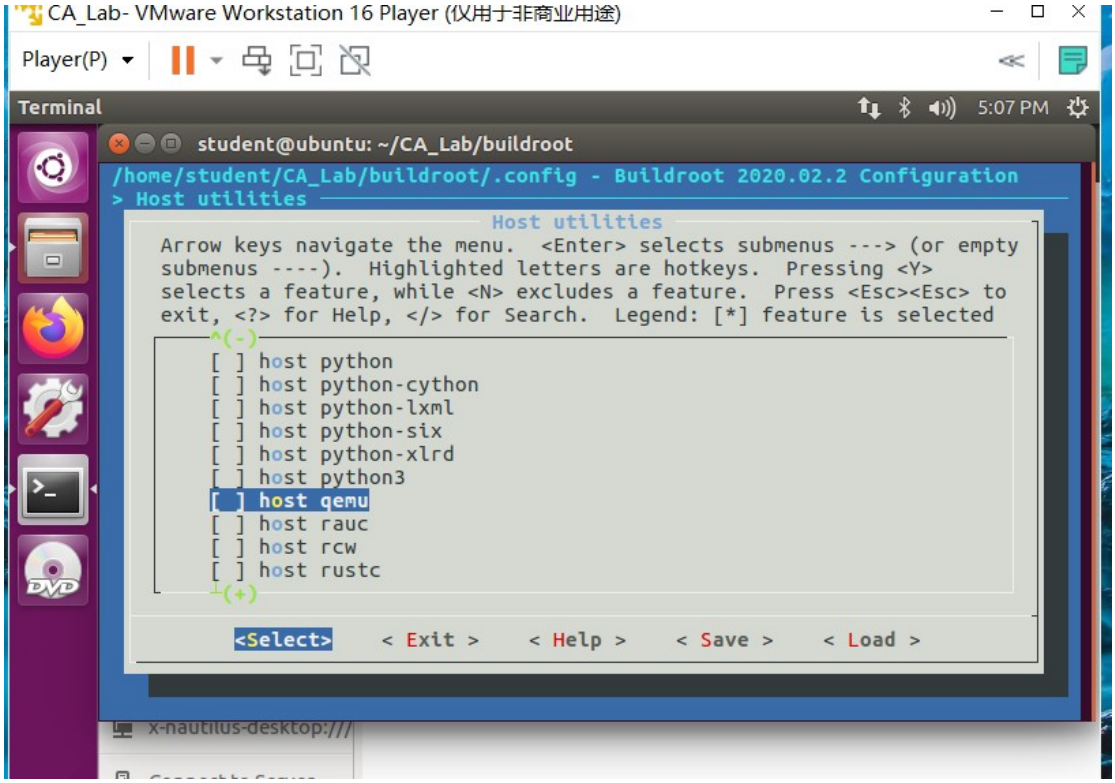


4. 在Filesystem images界面， 将ext2/3/4 variant 设置为ext4,并将大小改为500M

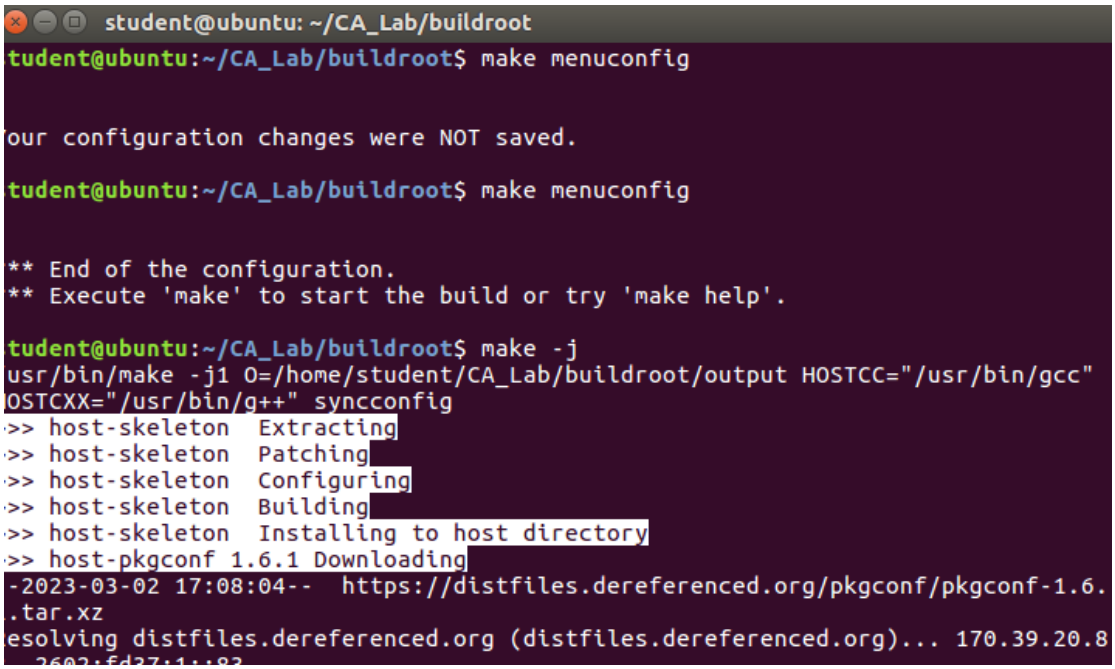



5. 在Host utilities界面取消勾选‘host qemu’




6. 保存并退出，一定要记得退出前保存，最开始忘了保存然后重新编译了一遍（悲）。

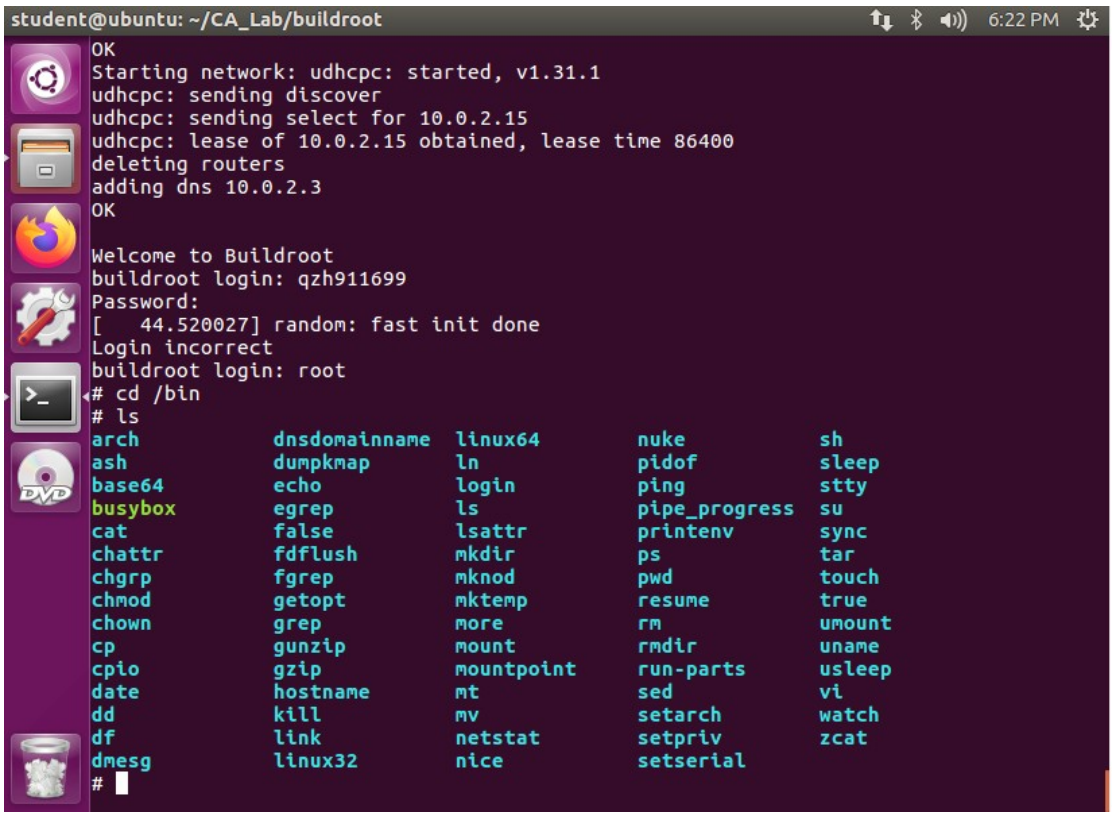



7. 配置完成后在 Buildroot 项目目录下运行指令：

```
$qemu-system-riscv64 \
    -M virt -nographic \
   -bios output/images/fw_jump.elf \
   -kernel output/images/Image \
```

```
-append "root=/dev/vda ro" \
-drive file=output/images/rootfs.ext4,format=raw,id=hd0 \
-device virtio-blk-device,drive=hd0 \
-netdev user,id=net0 -device virtio-net-device,netdev=net0
```

8. 可以使用 QEMU 启用 RISC-V Linux 系统，使用会员身份登入，查看系统相关信息：

使用 poweroff 可以关闭该操作系统。

```
cpio            gzip            mountpoint      run-parts       usleep
date            hostname        mt              sed             vi
dd              kill            mv              setarch         watch
df              link            netstat         setpriv         zcat
dmesg           linux32         nice            setserial
# poweroff
# Stopping network: OK
Saving random seed: [  218.114591] random: dd: uninitialized urandom read (512 byt
es read)
OK
Stopping klogd: OK
Stopping syslogd: OK
umount: devtmpfs busy - remounted read-only
[  218.277315] EXT4-fs (vda): re-mounted. Opts: (null)
The system is going down NOW!
Sent SIGTERM to all processes
Sent SIGKILL to all processes
Requesting system poweroff
[  220.291362] reboot: Power down
student@ubuntu:~/CA_Lab/buildroot$
```

（2）向 QEMU 上的 Linux 系统转移文件

方法一：使用 Mount

a. 首先在主机中使用工具链实现可执行文件：

```
$ riscv64-unknown-linux-gnu-gcc -static -o hello hello.c
```

这里的工具链采用的是 buildroot 为用户在主机中下载的工具链，也是将 c 语言文件编译为 riscv 的可执行文件。此处的工具链是 glibc 版本，生成的 C 程序执行在 Linux 系统中，而上一节的工具栏 newlib 是执行在裸机上。

```
student@ubuntu:~$ cd Desktop
student@ubuntu:~/Desktop$ riscv64-unknown-linux-gnu-gcc -static -o hello hello.c
student@ubuntu:~/Desktop$ ls
hello  hello.c  riscv-probe  text.txt
student@ubuntu:~/Desktop$
```

b. 在 Buildroot 制作的文件系统路径/output/images/下，建立一个新文件夹 tmpfs：

```
$ cd ./output/images
$ mkdir tmpfs
```

```
student@ubuntu:~/CA_Lab/buildroot$ cd ./output/images
student@ubuntu:~/CA_Lab/buildroot/output/images$ mkdir tmpfs
student@ubuntu:~/CA_Lab/buildroot/output/images$ sudo mount -t ext4 ./rootfs.ext4 ./tmpfs
-o loop
[sudo] password for student:
student@ubuntu:~/CA_Lab/buildroot/output/images$ sudo cp -r ~/Desktop/hello ./tmpfs/root
student@ubuntu:~/CA_Lab/buildroot/output/images$ sudo umount ./tmpfs
student@ubuntu:~/CA_Lab/buildroot/output/images$ cd
student@ubuntu:~$ cd CA_Lab
student@ubuntu:~/CA_Lab$ cd buildroot
student@ubuntu:~/CA_Lab/buildroot$ qemu-system-riscv64 \
>      -M virt -nographic \
>     -bios output/images/fw_jump.elf \
>     -kernel output/images/Image \
>     -append "root=/dev/vda ro" \
>     -drive file=output/images/rootfs.ext4,format=raw,id=hd0 \
>     -device virtio-blk-device,drive=hd0 \
>     -netdev user,id=net0 -device virtio-net-device,netdev=net0

OpenSBI v0.5 (Mar  2 2023 18:01:45)
```

c. 现在需要将主机中的 hello 可执行文件传递到 QEMU 的 Linux 系统中，在 QEMU 的 Linux 系统关闭的状态下使用：

```
$ sudo mount -t ext4 ./rootfs.ext4 ./tmpfs -o loop
```

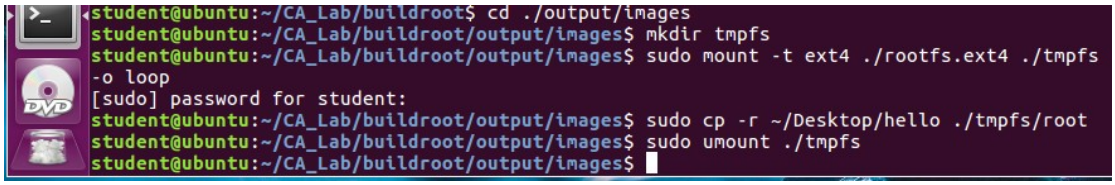
将 rootfs.ext4 文件系统挂载到文件夹 tmpfs 下，访问 tmpfs 文件时相当于访存 rootfs.ext4 文件系统。接着使用：

```
$ sudo cp -r ~/Desktop/hello ./tmpfs/root
```

使用 cp 将桌面上的 hello 文件复制到 tmpfs 子路径下，成功将主机编译完成的文件转移到 Linux 镜像中。最后：

```
$ sudo umount ./tmpfs
```

解除挂载




方法二：使用 ftp 传输

a. 在 Ubuntu 中启用 ftp 服务：

```
$ sudo apt install vsftpd
$ sudo service vsftpd restart
```

b. QEMU Linux 启动 lftp，然后输入密码

```
# lftp 10.0.2.3 -u student
```

c. 建立成功后，可以自由传输数据。get 和 put 进行文件传输，mirror 命令实现文件夹传输。

d.

```
# rm hello.c
# lftp 10.0.2.3 -u student
Password:
lftp student@10.0.2.3:~> cd Desktop
cd ok, cwd=/home/student/Desktop
lftp student@10.0.2.3:~/Desktop> !ls
lftp student@10.0.2.3:~/Desktop> !ls
lftp student@10.0.2.3:~/Desktop> ls
-rwxrwxr-x    1 1000     1000      4675808 Mar 02 18:24 hello
-rwxrw-rw-    1 1000     1000           61 Feb 24 05:22 hello.c
drwxrwxr-x    7 1000     1000         4096 Feb 26 03:47 riscv-probe
-rw-rw-r--    1 1000     1000          138 Feb 27 04:17 text.txt
lftp student@10.0.2.3:~/Desktop> get hello
4675808 bytes transferred
lftp student@10.0.2.3:~/Desktop> !ls
hello
```

（3）在 QEMU 上的 Linux 系统运行编译后的 C 程序

回到 buildroot 路径下，打开 QEMU 虚拟机，打开 root 路径，可以看见有 hello 文件，直接执行。

```
Welcome to Buildroot
buildroot login: root
# cd /root
# ls
hello
# hello
-sh: hello: not found
# ./hell[   60.121292] random: fast init done

-sh: ./hel: not found
# ./hello
hello
# ./hello
hello
#
```

3、实验分析和总结

（1）.本实验在 Linux 虚拟机上运行了一个 riscv 架构的 Linux 镜像，首先下载了 Buildroot 工具，用于生成各类 Linux 组件，在图形化配置界面配置完需要生成的目标机器标准并完成对 Buildroot 的编译之后，就通过 QEMU 来打开 Buildroot 生成的 RISCV 镜像。可以发现这里与实验二的不同之处在于此处打开的虚拟机是由 Buildroo 生成的 RISCV-Linux 系统，而实验二中的则是由 QEMU 自带的 spike 裸机。在本实验中的虚拟机带有一些基本的软件设施如与主机的文件传输 lftp 软件。

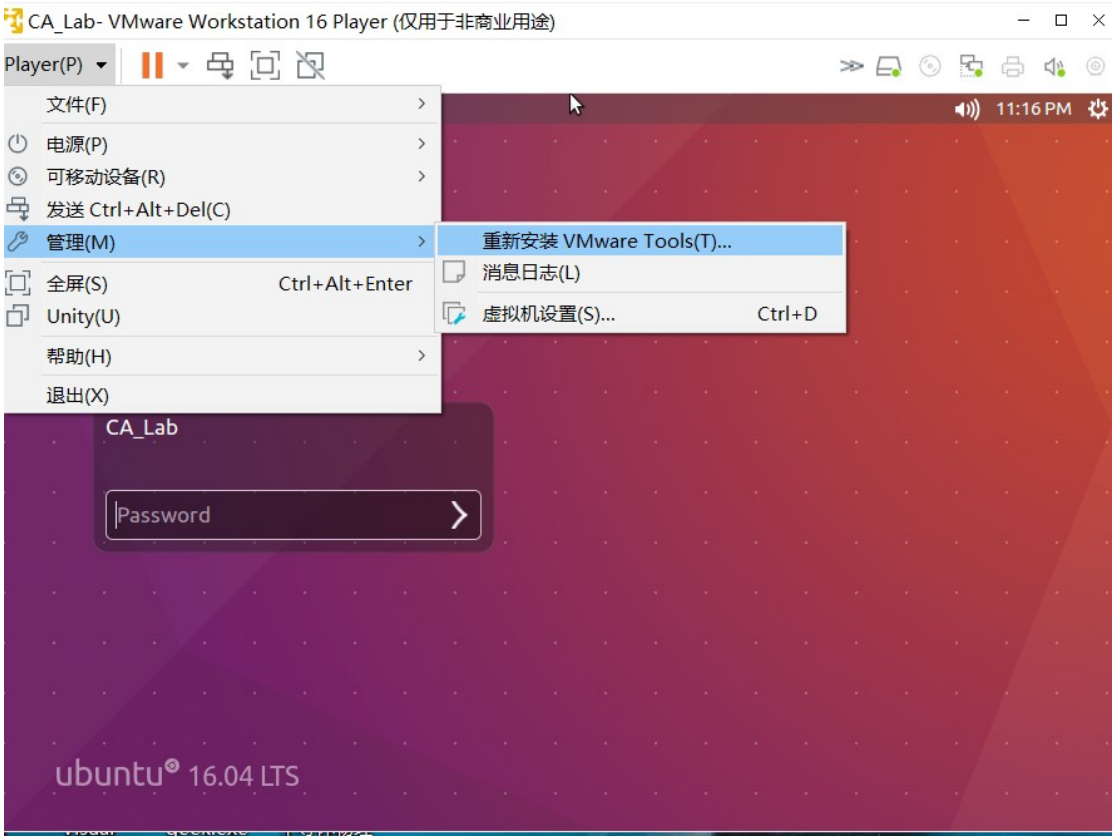
（2）.由于 buildroot 主要用于小型嵌入式设备的 Linux 架构，所以在生成的虚拟机上是没有编译工具的，需要将 c 语言文件在主机（Host）中完成编译后移动到虚拟机中执行。可以通过使用 Mount 挂载，将 QEMU 虚拟的系统的文件夹关联在(Host)上；也可以采用 lftp 工具，实现主机和虚拟机的问价传输。

（3）.在程序运行上，与实验二一样，直接在命令行执行可执行程序即可。

4、实验收获、存在问题、改进措施或建议等

实验收获：

（1）. 在实验中遇到了无法在主机和 Ubuntu 虚拟机之间复制粘贴和传输文件的情况，最后的解决方法是重新配置 VMware TOOL：




（2）. 关于交叉工具链：编译 c 语言的过程需要经过预处理、编译、汇编、链接一系列步骤，每个步骤都有对应的工具完成，前一个工具的输出是后一个工具的输入，整个工具形成一个工具链。但是在实际的嵌入式开发中，嵌入式设备的资源有限，多没有安装工具链，所以需要将 c 语言编译完成之后传输到嵌入式设备中，这导致需要用工具链完成对其他架构的程序编译，比如本实验中在 linux 系统中编译一个 RISCV 的程序，这就需要使用交叉工具链。

在实验二中使用的交叉工具链是 newlib 工具链，是一种针对裸机的工具链，而本实验的 glibc 工具链则是实现对 Linux 系统的工具链。同样适用于 Linux 系统的工具链还有 uClibc，eglibc,Musl-libc 等等，其区别主要在于 glibc 功能更为齐全，但是更加臃肿。而其他几个都主要面向嵌入式设备，更加轻量级。

此外，除了对不同架构的系统工具链不同之外，对于 windows 系统和 Linux 系统而言，两个系统中的 gcc 工具链也是不同的，在 linux 系统中编译好的可执行文件在 windows 系统下同样无法运行。

（3）. 两种传输文件的方法：

i. 使用 Mount 挂载

1、 首先将外设文件系统挂在 linux 文件下

2、 将外设当做 Linux 文件夹下的一个子目录操作

3、 完成操作之后接触绑定。

ii. 使用 ftp 传输

1、将外设通过 FTP 协议链接到主机上，之后可以直接访问到主机的文件。更加方便快捷，且不需要外设关闭。

存在问题：

（1）.对各种配置语句不太理解

（2）.对不同架构和系统下的工具链不太了解。

改进措施和建议：

（1）.在 csdn 上搜索相关的文章

（2）.与之前的实验相互对比学习。