RISCV-QEMU 上运行 linux 系统实验

一．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。

二．实验步骤

（1）使用 Buildroot 搭建 Linux 系统的配置和编译过程

下载：使用以下命令克隆 Buildroot 的最新版本：

git clone https://github.com/buildroot/buildroot

下载完成后如下图

```
student@ubuntu:~$ git clone https://github.com/buildroot/buildroot
Cloning into 'buildroot'...
remote: Enumerating objects: 499833, done.
remote: Counting objects: 100% (260/260), done.
remote: Compressing objects: 100% (104/104), done.
remote: Total 499833 (delta 174), reused 238 (delta 156), pack-reused 499573
Receiving objects: 100% (499833/499833), 160.16 MiB | 10.90 MiB/s, done.
Resolving deltas: 100% (329226/329226), done.
Checking connectivity... done.
student@ubuntu:~$
```

切换分支后：

$ cd buildroot

$git checkout 2f7183d13133f2ded97fee273bd0cbed10226e4e

编译所需要的绝大多数工具默认已经安装完成，此处文字提示你可以创建其他分支

```
student@ubuntu:~$ cd buildroot
student@ubuntu:~/buildroot$ git checkout 2f7183d13133f2ded97fee273bd0cbed10226e4
e
Note: checking out '2f7183d13133f2ded97fee273bd0cbed10226e4e'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 2f7183d... Update for 2020.02.2
student@ubuntu:~/buildroot$
```

使用以下命令安装 ncurses5 库，从而可以使用 buildroot 的图形化配置界面：

sudo apt install libncurses5-dev

输入命令后弹出密码输入，期间仍然不会显示数字，正常输入即可（如下图）

```
student@ubuntu:~$ sudo apt install libncurses5-dev
[sudo] password for student:
Reading package lists... Done
Building dependency tree
Reading state information... Done
libncurses5-dev is already the newest version (6.0+20160213-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 27 not upgraded.
student@ubuntu:~$
```

配置：（1）默认配置：configs 文件夹下包含了软件为各种不同架构和使用场景所生成的默 认配置，Buildroot 就是通过一系列配置文件，使用自动化脚本编译生成符合用户要求的文件。导入 buildroot 目录，并输入以下命令导入针对 RISC-V64 架构的默认配置文件：

make qemu_riscv64_virt_defconfig

```
student@ubuntu:~$ cd buildroot
student@ubuntu:~/buildroot$ make qemu_riscv64_virt_defconfig
make qemu_riscv64_virt_defconfig: command not found
student@ubuntu:~/buildroot$
```

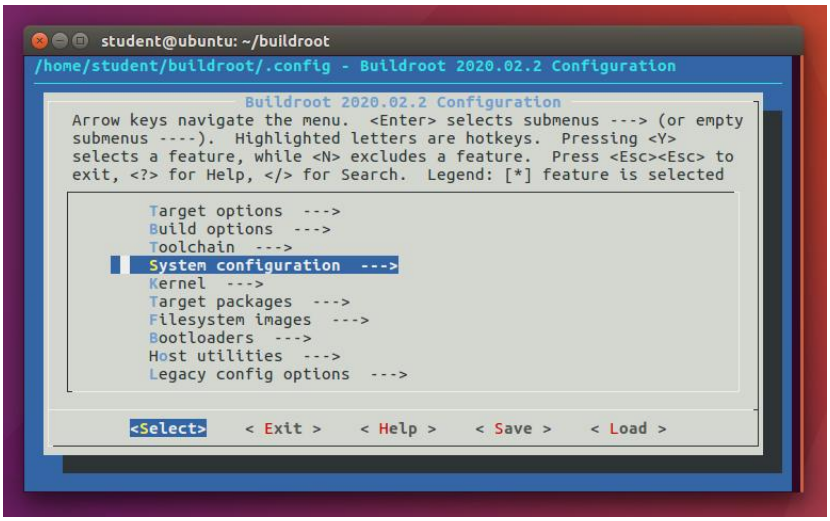
直接从 word 文档中复制该命令时会出现上图的错误，手动输入之后可以解决

下图中可在最后一行找到生成的 config 文件：

```
student@ubuntu:~/buildroot$ make qemu_riscv64_virt_defconfig
mkdir -p /home/student/buildroot/output/build/buildroot-config/lxdialog
PKG_CONFIG_PATH="" make CC="/usr/bin/gcc" HOSTCC="/usr/bin/gcc" \
    obj=/home/student/buildroot/output/build/buildroot-config -C support/kconfig
 -f Makefile.br conf
/usr/bin/gcc -D_GNU_SOURCE -DCURSES_LOC="<ncurses.h>" -DLOCALE  -I/home/student/
buildroot/output/build/buildroot-config -DCONFIG_=\"\"  -MM *.c > /home/student/
buildroot/output/build/buildroot-config/.depend 2>/dev/null || :
/usr/bin/gcc -D_GNU_SOURCE -DCURSES_LOC="<ncurses.h>" -DLOCALE  -I/home/student/
buildroot/output/build/buildroot-config -DCONFIG_=\"\"   -c conf.c -o /home/stud
ent/buildroot/output/build/buildroot-config/conf.o
/usr/bin/gcc -D_GNU_SOURCE -DCURSES_LOC="<ncurses.h>" -DLOCALE  -I/home/student/
buildroot/output/build/buildroot-config -DCONFIG_=\"\"  -I. -c /home/student/bui
ldroot/output/build/buildroot-config/zconf.tab.c -o /home/student/buildroot/outp
ut/build/buildroot-config/zconf.tab.o
/usr/bin/gcc -D_GNU_SOURCE -DCURSES_LOC="<ncurses.h>" -DLOCALE  -I/home/student/
buildroot/output/build/buildroot-config -DCONFIG_=\"\"   /home/student/buildroot
/output/build/buildroot-config/conf.o /home/student/buildroot/output/build/build
root-config/zconf.tab.o  -o /home/student/buildroot/output/build/buildroot-confi
g/conf
rm /home/student/buildroot/output/build/buildroot-config/zconf.tab.c
#
# configuration written to /home/student/buildroot/.config
#
student@ubuntu:~/buildroot$
```

（2）图形化配置：

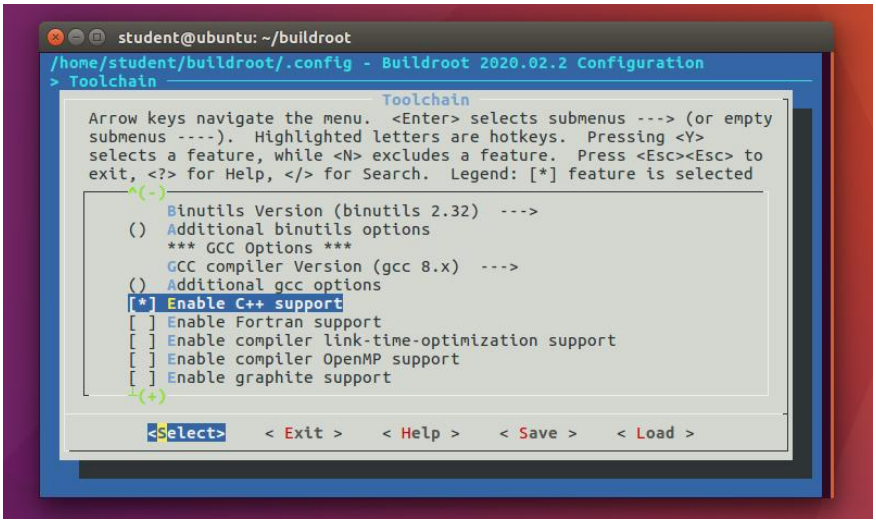
在 buildroot 目录下使用命令 make menuconfig 可以启动图形化配置界面，如下图



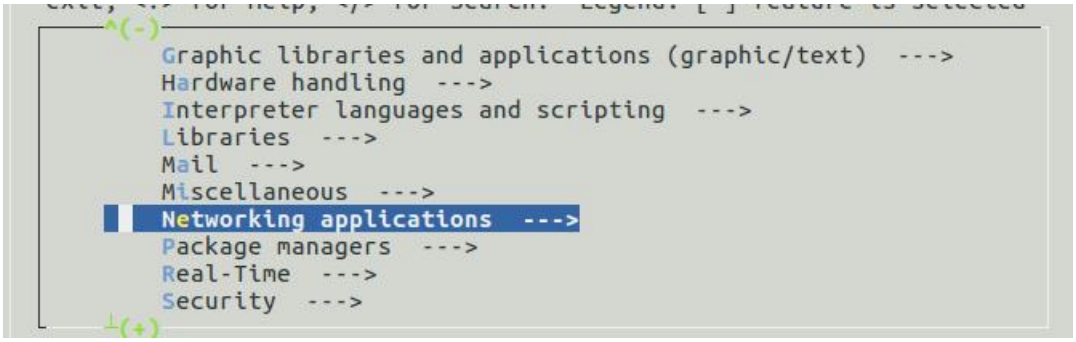
在上图中按上下方向键选择 Toolchain，然后选择 Enable C++ Support 选项

（Y 键可以选择选项，N 键取消选项，回车键进入下一级目录，/键可以进行关键字搜索，左右方向键可在底部的选项中选择）

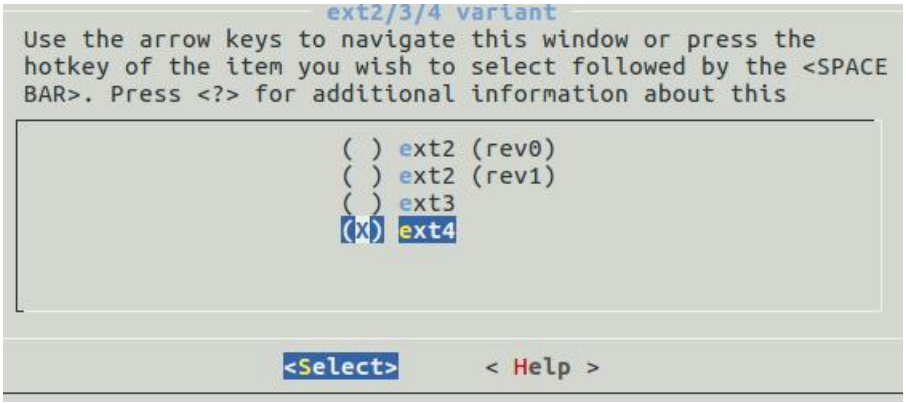
选择结果如下图



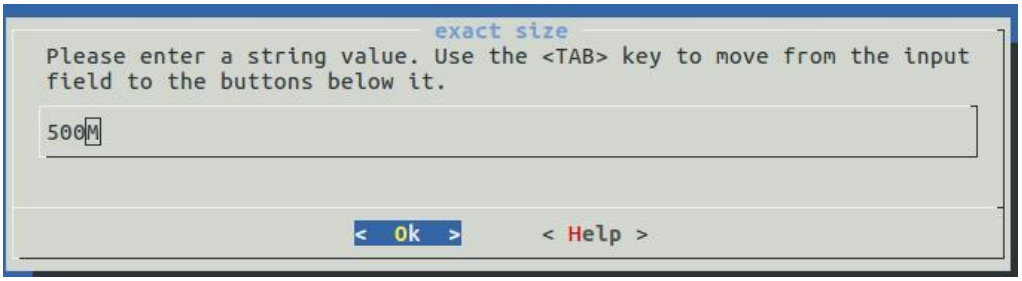
在 Target Packages 中选择 Networking applications 目录，并将其中的 lftp 工具选中，



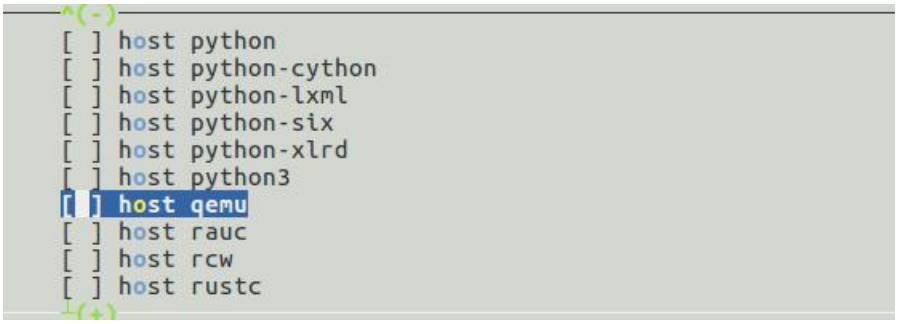
在 Filesystem images 中选择 ex2/3/4 filesystem 并将 ex2/3/4 variant 设置为 ext4，如下图（即使用 ext4 类型的文件系统）



将 exact size 修改为 500M：（如下图）



将 Host utilities 下的 host qemu 取消（不需要再安装一次 qemu）：



（3）编译：在配置完成之后，切换到 buildroot 目录下使用 make 命令进行编译：

```
student@ubuntu: ~/buildroot
student@ubuntu:~$ cd buildroot
student@ubuntu:~/buildroot$ make
>>> linux-headers 5.1.12 Downloading
--2023-03-02 16:14:10--  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.1.
12.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.77.176, 2a04:4e42:12::432
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.77.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 106265072 (101M) [application/x-xz]
Saving to: '/home/student/buildroot/output/build/.linux-5.1.12.tar.xz.OESiOx/out
put'

put/build/.linux-5.    1%[                    ]   1.91M   104KB/s    eta 9m 39s
```

```
tudent/buildroot/output/build/buildroot-fs/tar/fakeroot
chmod a+x /home/student/buildroot/output/build/buildroot-fs/tar/fakeroot
PATH="/home/student/buildroot/output/host/bin:/home/student/buildroot/output/hos
t/sbin:/opt/riscv-qemu/bin:/opt/riscv-newlib/bin:/opt/riscv-linux/bin:/home/stud
ent/bin:/home/student/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/b
in:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin" FAKEROOTDONTTRYCHOWN=1 /hom
e/student/buildroot/output/host/bin/fakeroot -- /home/student/buildroot/output/b
uild/buildroot-fs/tar/fakeroot
rootdir=/home/student/buildroot/output/build/buildroot-fs/tar/target
table='/home/student/buildroot/output/build/buildroot-fs/full_devices_table.txt'
ln -snf /home/student/buildroot/output/host/riscv64-buildroot-linux-gnu/sysroot
/home/student/buildroot/output/staging
student@ubuntu:~/buildroot$
```

上图为编译完成界面

下图为 buildroot 目录下的子目录

```
student@ubuntu:~/buildroot$ ls
arch      Config.in         DEVELOPERS  linux           package  toolchain
board     Config.in.legacy  dl          Makefile        README   utils
boot      configs           docs        Makefile.legacy support
CHANGES   COPYING           fs          output          system
student@ubuntu:~/buildroot$
```

（4）使用 qemu 启动 riscv linux 系统：

在 buildroot 目录下使用以下命令启动镜像：

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

```
student@ubuntu:~/CA_Lab/buildroot$ sudo qemu-system-riscv64 \-M virt -nographic
\-bios output/images/fw_jump.elf \-kernel output/images/Image \-append "root=/de
v/vda ro" \-drive file=output/images/rootfs.ext4,format=raw,id=hd0 \-device virt
io-blk-device,drive=hd0 \-netdev user,id=net0 -device virtio-net-device,netdev=n
et0
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
            [command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-u user] file ...
student@ubuntu:~/CA_Lab/buildroot$
```

以上界面是输入命令之后分配系统资源的界面，使用 root 命令可以登入
以下为登入之后的界面

```
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-devi
ce,drive=hd0 -netdev user,id=net0 -device virtio-net-device,netdev=net0

OpenSBI v0.5 (Mar  3 2023 09:41:45)
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 0
```

使用 ls 可以查看文件目录

```
Welcome to Buildroot
buildroot login: [    1.264773] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link become
s ready
root
# cd /bin
# ls
arch            dnsdomainname  linux64        nuke           sh
[ 1112.804156] random: fast init done
ash             dumpkmap       ln             pidof          sleep
base64          echo           login          ping           stty
busybox         egrep          ls             pipe_progress  su
cat             false          lsattr         printenv       sync
chattr          fdflush        mkdir          ps             tar
chgrp           fgrep          mknod          pwd            touch
chmod           getopt         mktemp         resume         true
chown           grep           more           rm             umount
cp              gunzip         mount          rmdir          uname
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
#
```

使用 poweroff 可以关闭该操作系统，如下图：

```
# poweroff
# Stopping network: OK
Saving random seed: [ 1670.152122] random: dd: uninitialized urandom read (512 b
ytes read)
OK
Stopping klogd: OK
Stopping syslogd: OK
umount: devtmpfs busy - remounted read-only
[ 1670.436493] 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
[ 1672.464787] reboot: Power down
student@ubuntu:~/CA_Lab/buildroot$
```

（5）在 QEMU LINUX 上运行 C 程序：

方法一：使用 mount 挂载：首先使用以下命令，在 hello.c 对应的文件目录下将其转换为可执行文件：

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

该命令使用的工具链是 glibc 版本

然后将该文件移动到 buildroot 制作的文件路径./output/images 下，使用以下命令在对应的路径下先创建一个文件夹：

$ cd ./output/images

$ mkdir tmpfs

在保持 qemu 上的 linux 系统关闭的情况下，输入以下命令，可以完成 hello 文件的移动：

$ sudo mount -t ext4 ./rootfs.ext4 ./tmpfs -o loop
$ sudo cp -r ~/Desktop/hello ./tmpfs/root （蓝色部分是 hello 文件的地址）
$ sudo umount ./tmpfs
运行结束之后，重新打开 linux 系统，在其中输入./hello，就可以运行该程序：

```
Welcome to Buildroot
buildroot login: root
# ./hello
hello
#
```

方法二：使用 ftp 传输：（该传输方式不用关闭 linux 系统，较为方便）
在 ubuntu 端启用 ftp 服务，使用以下命令，如下图所示：
sudo apt install vsftpd
sudo service vsftpd restart

```
student@ubuntu:~$ sudo apt install vsftpd
[sudo] password for student:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  vsftpd
0 upgraded, 1 newly installed, 0 to remove and 27 not upgraded.
Need to get 115 kB of archives.
After this operation, 336 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 vsftpd amd64 3.0.3-3
ubuntu2 [115 kB]
```

在启动的 linux 中使用以下命令开起 lftp 和 Linux 的连接：
lftp 10.0.2.3 -u student
开启之后输入密码，可以输入对应的命令进行操作，如下图中的 cd 和 ls 命令：
lcd 和!ls 是当前 QEMU Linux 自身的路径移动和文件查看命令。使用 get 和 put 可以进行单个文件的传输，使用 mirror 命令则可以进行文件夹的传输。使用 exit 或者 quit 可以退出

```
Welcome to Buildroot
buildroot login: root
# lftp 10.0.2.3 -u student
[   12.942179] random: fast init done
Password:
lftp student@10.0.2.3:~> 123456
Unknown command `123456'.
lftp student@10.0.2.3:~> cd ~/Desktop/LAB2/
cd ok, cwd=/home/student/Desktop/LAB2
lftp student@10.0.2.3:~/Desktop/LAB2> LS
-rwxrwxr-x    1 1000     1000        10224 Feb 25 18:43 example
-rw-rw-r--    1 1000     1000          729 Feb 25 18:43 example.c
-rwxrwxr-x    1 1000     1000      4675808 Mar 03 10:16 hello
-rw-rw-r--    1 1000     1000           97 Feb 24 10:15 hello.c
lftp student@10.0.2.3:~/Desktop/LAB2>
```

（使用平头哥提供的工具可以在 qemu 上配置 linux 系统。首先下载工具链并解压，然后下载并编译平头哥的 buildroot，之后利用 csky-qemu 工具启动，就可以在 qemu 上运行 linux 系统；也可以全部使用平头哥制作完成的文件进行解压。）

三．实验分析

（1）实验过程中复制代码到命令窗口中时，会出现命令未找到的错误提示，此种问题可以通过手动输入代码或者删除某些空格来实现，因为代码在从 word 文档中复制到 ubuntu 的过程中，空格字符会出错。
（2）需要在默认配置和图形化配置全部完成之后再进行编译，并且配置所处的文件夹需要保持一致，否则可能会出现“no such file or directory”的错误（比如图形化配置中 ext4 文件就没有下载）

（3）编译过程中会出现 recipe for target ‘xxxxx‘ failed 的错误，该错误说明 build 文件已经生成了，需要使用命令 make clean 清除，然后再进行编译。

（4）在进行图形化配置的过程中，在选项中使用 Y 勾选后需要在底部的选项中选择 save 进行保存。

（5）在使用 mount 挂载方法时，关闭 linux 系统的前提下输入了三行代码，第一行：

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

将文件系统 rootfs.ext4 挂载到文件夹 tmpfs 下，此时访问 tmpfs 就相当于在访问 rootfs.ext4；

（此处 rootfs.ext4 即为前面步骤中图形化配置中配置的文件系统）

第二行：sudo cp -r ~/Desktop/hello ./tmpfs/root 使用 cp 命令将 hello 文件复制到了 tmpfs 的子路径下；

（~/Desktop/为 hello 文件的位置）

第三行：sudo umount ./tmpfs 解除了挂载状态

在第二行命令执行之后，文件实际上已经完成了转移。

四．实验总结

（1）Buildroot 是一个用交叉工具链来创建嵌入式 Linux 系统的开源框架。

下载之后查看目录如下。buildroot 本身使用和生成的文件夹包含如下：

-output：存放 buildroot 基本上所有的编译输出文件

-dl: 源码和软件包的下载位置

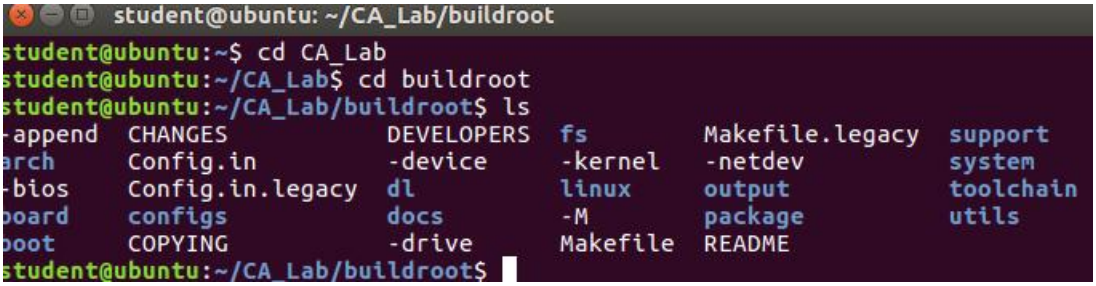
-fs: 文件系统源码

-configs：各类参数配置文件

-arch：CPU 架构的相关配置文件

-linux：内建构建脚本

-docs：参考文档存放位置



（2）本实验利用 buildroot 工具，在 ubuntu 中搭建出一个 RISC-V Linux 系统，并可以运行 C 程序。搭建过程中需要编译出内核镜像，制作相关的程序、初始化引导系统和正确的文件系统，buildroot 就是完成这些工作的集成工具，它可以生成交叉工具链、内核镜像、根文件系统、引导程序、目标软件包（各类用户选择的），这些生成的内容可以在图形化配置界面中进行详细地选择与配置。

（3）buildroot 生成的各部分模块的功能描述：在一个嵌入式设备上启动 linux 系统时，引导程序会在上电后先获得控制权，它可以初始化硬件设备、建立内存空间映射图，为操作系统的运行准备必要的环境。完成环境设置之后，加载内核映像到内存，在特定的入口程序处完成内核引导并挂载根文件系统，此后系统发起首个 init 用户进程，依据运行级别启动守护进程，完成系统的初始化工作并最终为用户提供登录窗口。

（4）在 ubuntu 主机中的 c 程序需要在编译之后才能在 linux 系统中运行，但 Linux 中没有 gcc 编译器，所以需要 C 文件先在主机中编译完成之后再传输到 qemu linux 系统中。传输的方法有两种，一种是 mount 挂载法，一种是 ftp 传输，前者需要在关不 linux 的状态下转移 C 文件，后者的优势则是可以不用在关闭 linux 系统的情况下进行操作，更加自由灵活。

五．实验心得

（1）在输入命令时，需要转到对应能够生效的目录下，避免因为命令执行位置而出现的错误。

（2）复制的代码在输粘贴到命令窗口中的时候，会出现命令不存在的错误，此时可以选择重新输入，或者将复制

的代码中多余的空格删去，避免复制过程中非法字符的出现。