

计算机系统综合设计实验报告

Ubuntu 版本：18.04(老师发的指导书上说用 Ubuntu20.04 出现了问题)

班级

计算机91

姓名

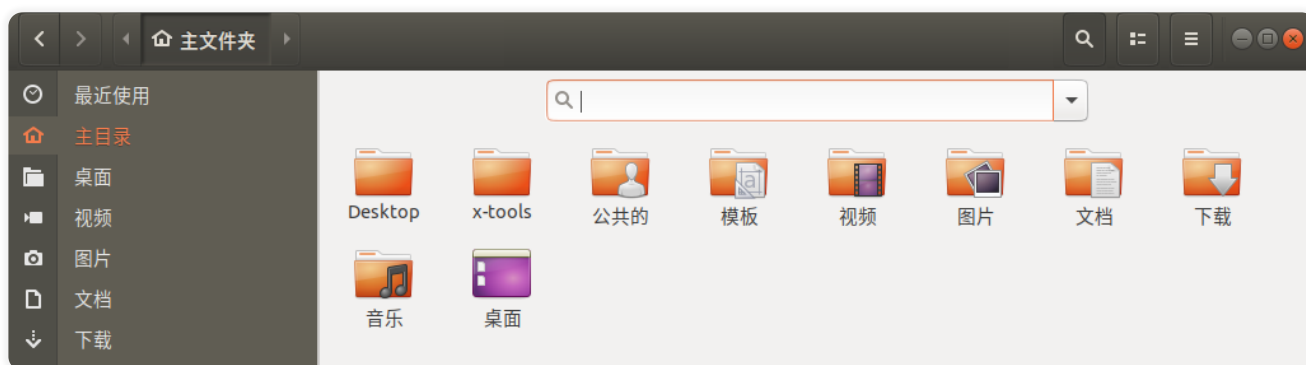
刘青帅

学号

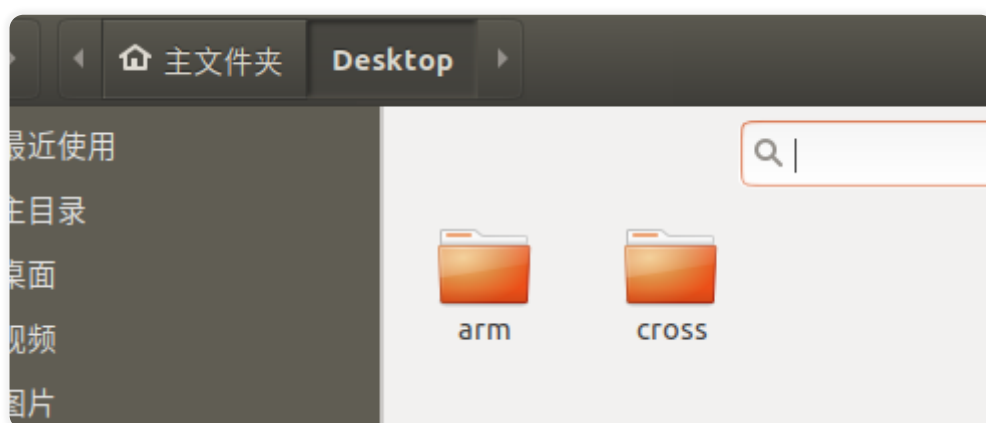
2191211634

说明：如果按照下面的一步一步来，应该不会出现问題(经王姐检验，是这样的)

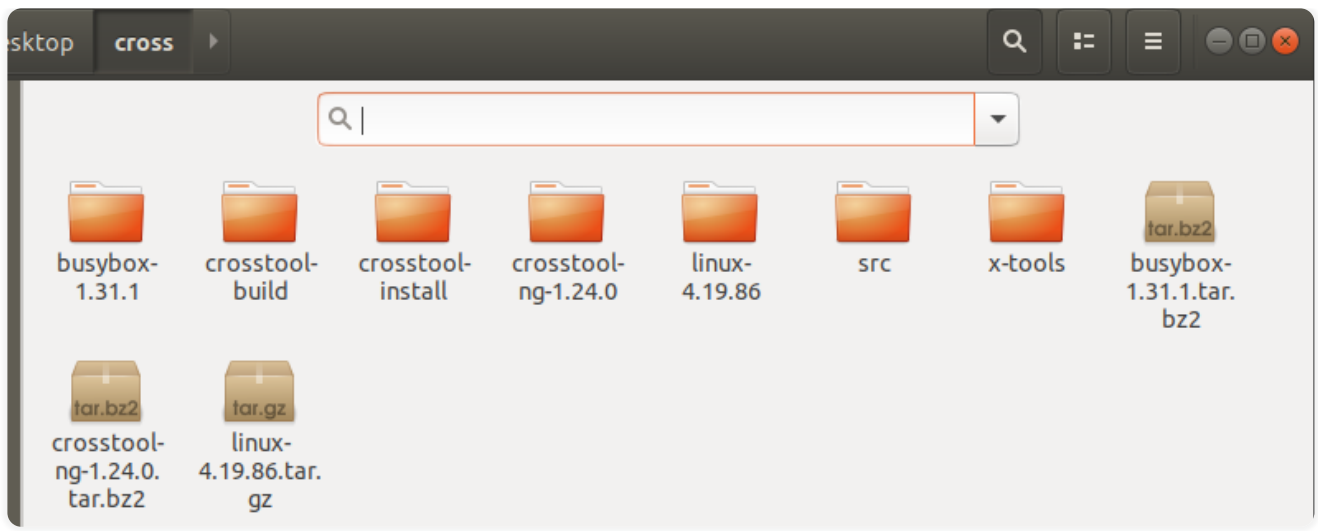
我新建了一个 Desktop 文件夹，Desktop和桌面是两个独立的文件夹，你如果看到目录有桌面这样的字眼，这是我之前做的，截图忘改了。(这里建议用英文路径，用中文的桌面其实也没问题，就是后面修改一个路径的时候 configmenu 无法输入中文，需要 vim 输入中文)



Desktop 目录下新建 cross 文件夹和 arm 文件夹



cross 的目录如下 (先看一下就行，防止后面乱，你可以回看一下)



H2 一、面向飞腾处理器编译Linux内核和基本工具

提前安装下面东西，防止出错

```
1 | sudo apt-get install make qemu-system-arm gcc-arm-linux-gnueabi libncurses5-dev bison flex vim gcc g++ build-essential
```

H3 1.1实验目的

利用 QEMU 创建飞腾（ARM）架构计算机，在此之上编译一个基本的 Linux 操作系统

H3 1.2实验步骤

先下载好 busybox 和 linux 内核的源码，并解压

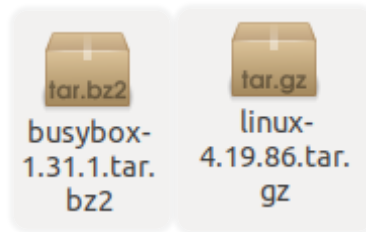
下载 busybox

```
1 | wget https://busybox.net/downloads/busybox-1.31.1.tar.bz2
```

谷歌搜索 linux-4.19.86 自己下载

```
liuqingshuai@liuqingshuai-VirtualBox:~$ wget https://busybox.net/downloads/busybox-1.31.1.tar.bz2
--2022-06-04 12:00:20-- https://busybox.net/downloads/busybox-1.31.1.tar.bz2
正在解析主机 busybox.net (busybox.net)... 140.211.167.122
正在连接 busybox.net (busybox.net)|140.211.167.122|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度： 2430221 (2.3M) [application/x-bzip2]
正在保存至：“busybox-1.31.1.tar.bz2”
```

下载并解压好之后(右键压缩的文件，手动提取到当前目录就行)

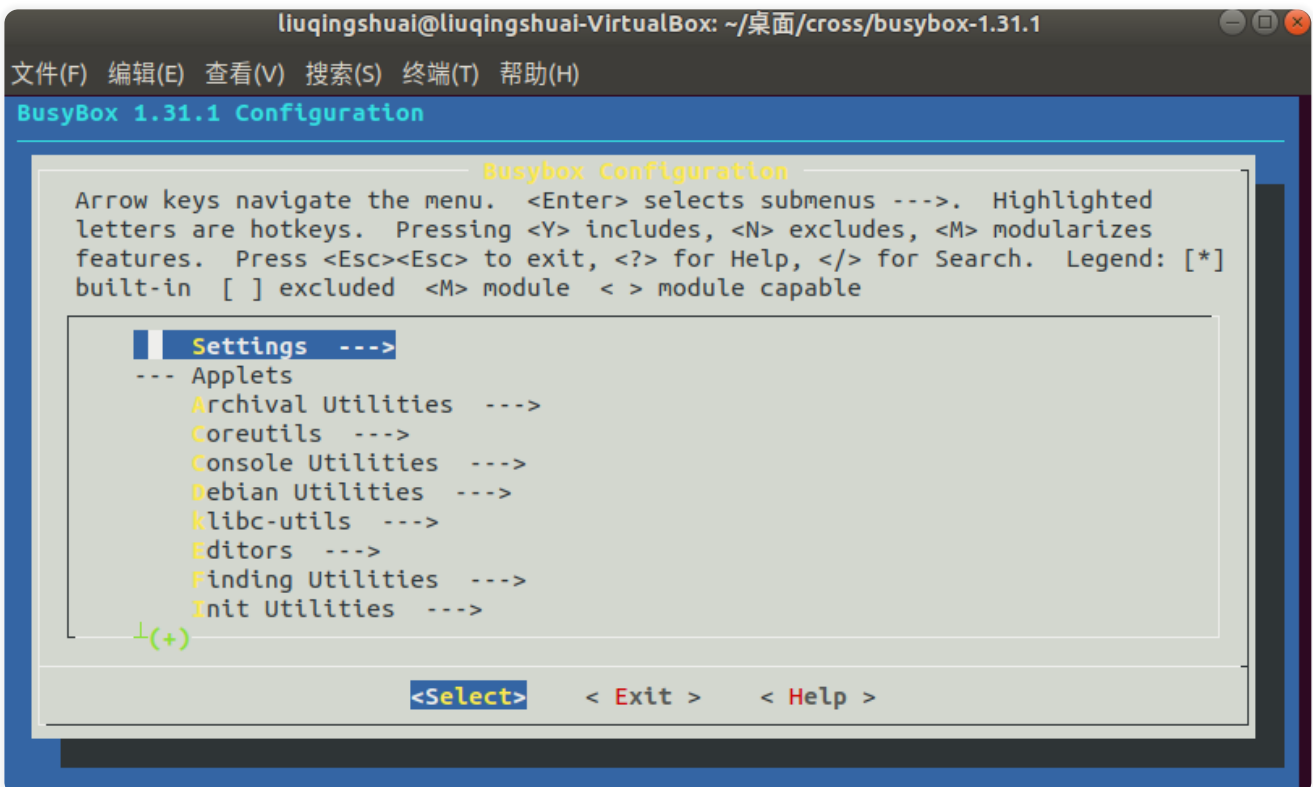


cd 到解压后的 busybox 的目录里面，执行下面三个指令，

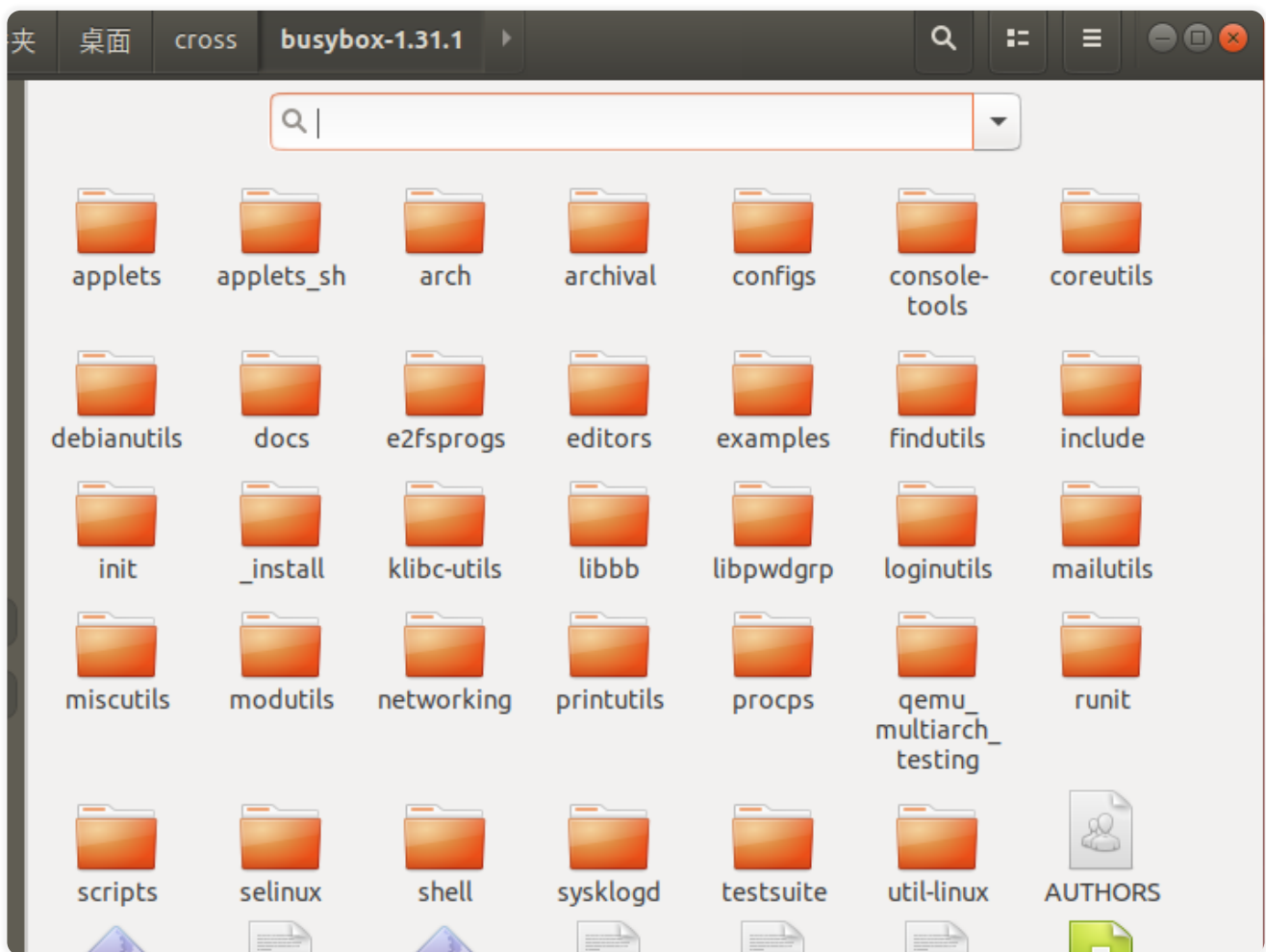
其中第一个出现menu只需要退出就行，里面已经勾选了默认的参数

上面是指导书上的说的，不要这样干，千万不要直接退出，要把下面的Build static binary勾选上!!!

```
1 make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- menuconfig
2 一定注意把这个选上 选中Settings→Build static binary(no shared libs) (NEW) 往下滑滑滑
3 make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-
4 make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- install
```



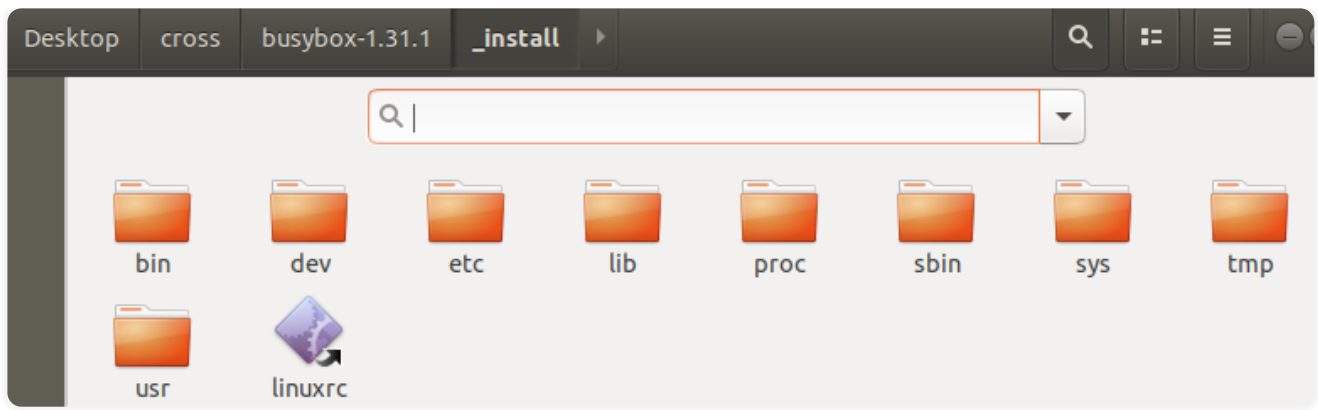
```
Settings
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >
^(-)
[ ] Support NSA Security Enhanced Linux (NEW)
[ ] Clean up all memory before exiting (usually not needed) (NEW)
[*] Support LOG_INFO level syslog messages (NEW)
--- Build Options
[*] Build static binary (no shared libs)
[ ] Force NOMMU build (NEW)
() Cross compiler prefix (NEW)
() Path to sysroot (NEW)
() Additional CFLAGS (NEW)
() Additional LDFLAGS (NEW)
L(+)
```



上面3个make指令完毕后，在 `_install` 目录下生成一些文件,创建下列文件夹备用： `etc` `proc` `sys` `tmp` `dev` `lib`

```
1 | mkdir etc proc sys tmp dev lib

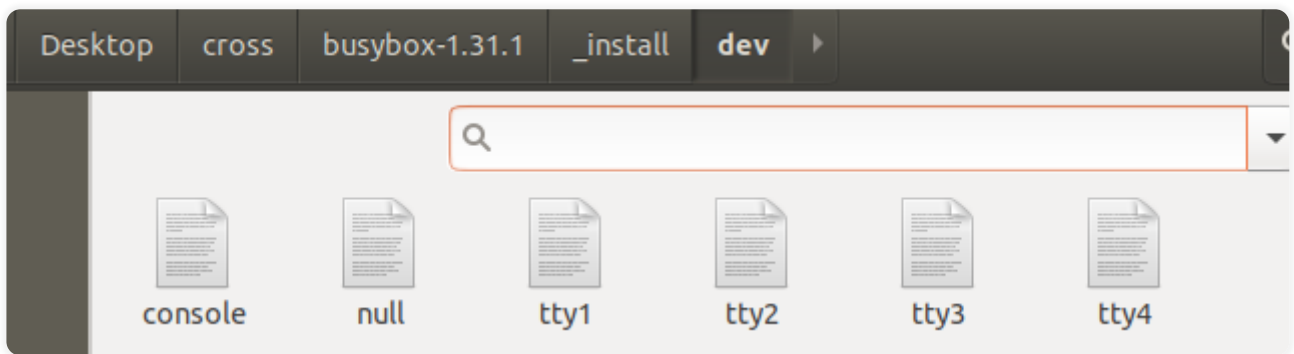
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/busybox-1.31.1/_install$ mkdir etc proc sys tmp dev lib
```



dev 目录下静态创建如下节点：

```
1 sudo mknod -m 666 tty1 c 4 1
2 sudo mknod -m 666 tty2 c 4 2
3 sudo mknod -m 666 tty3 c 4 3
4 sudo mknod -m 666 tty4 c 4 4
5 sudo mknod -m 666 console c 5 1
6 sudo mknod -m 666 null c 1 3
```

```
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/busybox-1.31.1/_install/dev$ sudo mknod -m 666 tty1 c 4 1
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/busybox-1.31.1/_install/dev$ sudo mknod -m 666 tty2 c 4 2
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/busybox-1.31.1/_install/dev$ sudo mknod -m 666 tty3 c 4 3
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/busybox-1.31.1/_install/dev$ sudo mknod -m 666 tty4 c 4 4
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/busybox-1.31.1/_install/dev$ sudo mknod -m 666 console c 5 1
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/busybox-1.31.1/_install/dev$ sudo mknod -m 666 null c 1 3
```



向 `_install/etc` 目录新建 `inittab` 文件、`fstab` 文件(这是二进制文件，没有后缀)

```
1 touch inittab fstab
```

使用 `vim` 写入文件

`inittab` 文件如下



```
1  ::sysinit:/etc/init.d/rcS
2  ::askfirst:/bin/sh
3  ::ctrlaltdel:/sbin/reboot
4  ::shutdown:/sbin/swapoff -a
5  ::shutdown:/bin/umount -a -r
6  ::restart:/sbin/init
7  tty2::askfirst:/bin/sh
8  tty3::askfirst:/bin/sh
9  tty4::askfirst:/bin/sh
```

fstab 文件如下



```
1  #device mount-point type option dump fsck order
2  proc /proc proc defaults 0 0
3  temps /tmp rpoc defaults 0 0
4  none /tmp ramfs defaults 0 0
5  sysfs /sys sysfs defaults 0 0
6  mdev /dev ramfs defaults 0 0
```

向 **etc** 目录新建 **init.d**文件夹，在 **init.d**文件夹 下新建 **rcS**文件



```
1  mkdir init.d
2  cd ./init.d
3  touch rcS
```

rcS 文件如下



```
1  mount -a
2  echo "/sbin/mdev" > /proc/sys/kernel/hotplug
3  /sbin/mdev -s
4  mount -a
```

然后一定要修改 **rcS** 文件的权限

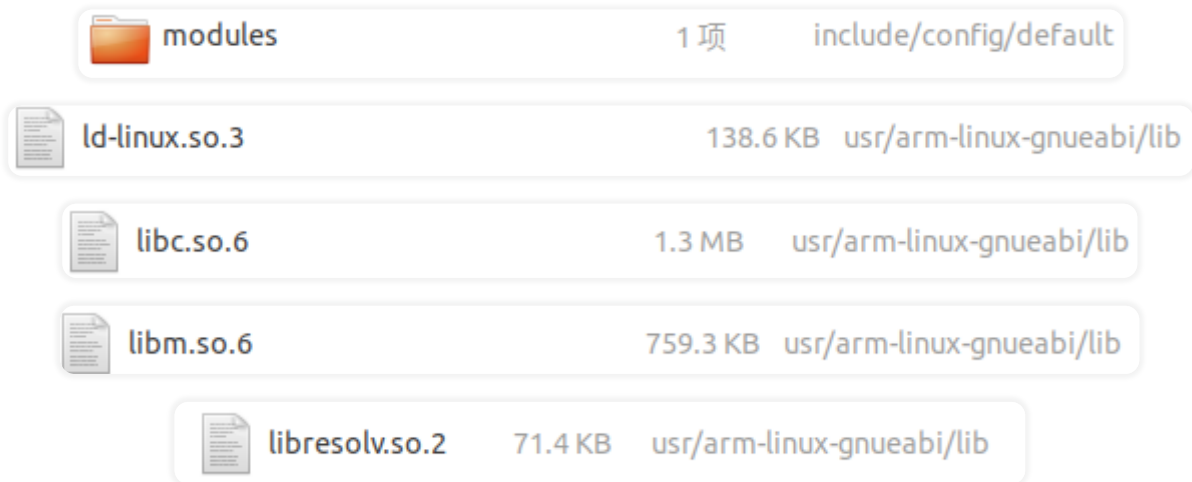


```
1  chmod +x rcS
```

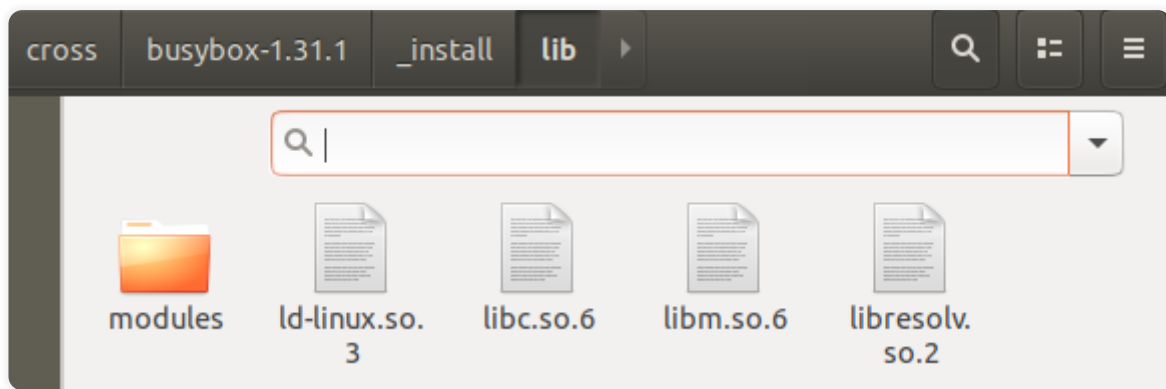
```
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/busybox-1.31.1/_install/etc/init.d$ chmod +x rcs
```

把依赖的库文件拷贝到 `_install/lib/` 目录下(如下)

`busybox` 目录下搜索 `modules` 文件夹, 全局搜索下面文件, 均拷贝到 `_install/lib` 目录下



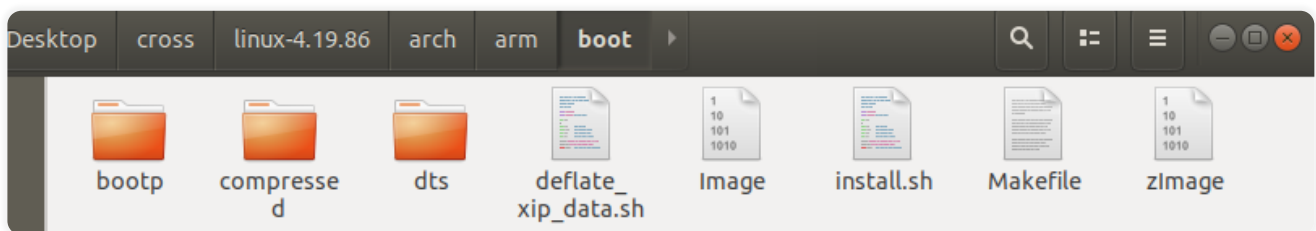
结果如下



接下来编译 `linux` 内核 (在解压的 `linux` 目录下 `make`) , 第二步时间很长, 你可以偷着乐

```
1 make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- vexpress_defconfig
2 make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-
```

此时 `boot` 目录如下



再进行如下步骤 (在 `linux` 目录下make)

```
1 make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- modules_install  
INSTALL_MOD_PATH="/home/liuqingshuai/Desktop/cross/busybox-1.31.1/_install/"
```

```
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/linux-4.19.86$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- modules_install INSTALL_MOD_PATH="/home/liuqingshuai/Desktop/cross/busybox-1.31.1/_install/"  
INSTALL crypto/drbg.ko  
INSTALL crypto/echainiv.ko  
INSTALL crypto/hmac.ko  
INSTALL crypto/jitterentropy_rng.ko  
INSTALL crypto/sha256_generic.ko  
INSTALL drivers/video/backlight/lcd.ko  
DEPMOD 4.19.86
```

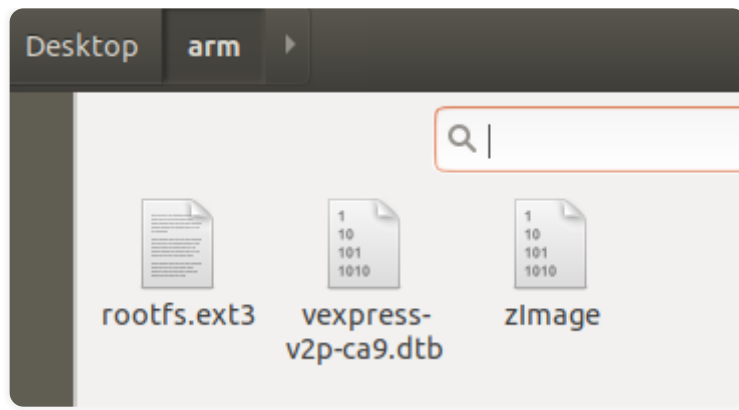
Desktop 上创建一个 arm 文件夹，然后在 arm 目录下执行下面步骤

```
1 dd if=/dev/zero of=rootfs.ext3 bs=1M count=32  
2 mkfs.ext3 rootfs.ext3  
3 sudo mount -o loop rootfs.ext3 /tmp  
4 sudo cp -a /home/liuqingshuai/Desktop/cross/busybox-1.31.1/_install/* /tmp  
5 sudo umount /tmp
```

```
t3 bs=1M count=32  
记录了32+0 的读入  
记录了32+0 的写出  
33554432 bytes (34 MB, 32 MiB) copied, 0.0272756 s, 1.2 GB/s  
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/arm$ mkfs.ext3 rootfs.ext3  
mke2fs 1.44.1 (24-Mar-2018)  
丢弃设备块: 完成  
创建含有 32768 个块 (每块 1k) 和 8192 个inode的文件系统  
文件系统UUID: 8f20045a-9e88-4720-bd5a-d274ed1c826d  
超级块的备份存储于下列块:  
8193, 24577  
  
正在分配组表: 完成  
正在写入inode表: 完成  
创建日志 (4096 个块) 完成  
写入超级块和文件系统账户统计信息: 已完成  
  
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/arm$ sudo mount -o loop rootfs.ext3 /tmp  
[sudo] liuqingshuai 的密码:  
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/arm$ sudo cp -a /home/liuqingshuai/Desktop/cross/busybox-1.31.1/_install/* /tmp  
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/arm$ sudo umount /tmp  
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/arm$ S
```

将 /home/liuqingshuai/Desktop/cross/linux-4.19.86/arch/arm/boot 的 zImage 拷贝到 arm 目录下

将 /home/liuqingshuai/Desktop/cross/linux-4.19.86/arch/arm/boot/dts 的 vexpress-v2p-ca9.dtb 拷贝到 arm 目录下



在 arm 下执行如下命令

```
1 qemu-system-arm -M vexpress-a9 -kernel ./zImage -nographic -m 512M -smp 4 -sd ./rootfs.ext3 -dtb vexpress-v2p-ca9.dtb -append "init=/linuxrc root=/dev/mmcblk0 rw rootwait earlyprintk console=ttyAMA0"
```

```
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/arm$ qemu-system-arm -M vexpress-a9 -kernel ./zImage -nographic -m 512M -smp 4 -sd ./rootfs.ext3 -dtb vexpress-v2p-ca9.dtb -append "init=/linuxrc root=/dev/mmcblk0 rw rootwait earlyprintk console=ttyAMA0"
WARNING: Image format was not specified for './rootfs.ext3' and probing guessed raw.
        Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
        Specify the 'raw' format explicitly to remove the restrictions.
pulseaudio: set_sink_input_volume() failed
pulseaudio: Reason: Invalid argument
pulseaudio: set_sink_input_mute() failed
pulseaudio: Reason: Invalid argument
ALSA device list:
 #0: ARM AC'97 Interface PL041 rev0 at 0x10004000, irq 24
input: ImExPS/2 Generic Explorer Mouse as /devices/platform/smb@4000000/smb@4000000:motherboard/smb@4000000:motherboard:iopfpga@7,00000000/10007000.kmi/serio1/input/input2
EXT4-fs (mmcblk0): mounting ext3 file system using the ext4 subsystem
random: fast init done
EXT4-fs (mmcblk0): mounted filesystem with ordered data mode. Opts: (null)
VFS: Mounted root (ext3 filesystem) on device 179:0.
Freeing unused kernel memory: 1024K
Run /linuxrc as init process
random: crng init done
mount: mounting temps on /tmp failed: No such device

Please press Enter to activate this console.
```

用几个命令测试一下

```

Please press Enter to activate this console.
/bin/sh: can't access tty; job control turned off
/ # ls
bin          etc          linuxrc      proc         sys         usr
dev          lib          lost+found   sbin        tmp
/ # cd ./etc
/etc # ls -a
.           ..          fstab        init.d       inittab
/etc #

```

至此，实验完了

H3 1.3实验结果

本次实验利用 `qemu` 创建飞腾架构计算机，在此之上编译一个基本的 `Linux` 操作系统，通过 `Busybox` 构建了基本的系统命令。

H2 二、面向飞腾处理器的交叉编译环境

提前安装，防止出错

```

1 | sudo apt-get install autoconf automake libtool libncurses5-dev gperf texinfo
   | help2man gawk libtool-bin

```

H3 2.1实验目的

利用 `crosstool` 制作一个交叉编译工具链，使其能交叉编译c源文件，生成飞腾平台下的可执行文件

H3 2.2 实验步骤

下载 `crosstool` 压缩包然后压缩包右键提取一下

```

1 | wget http://crosstool-ng.org/download/crosstool-ng/crosstool-ng-1.24.0.tar.bz2

```

和 `crosstool-ng-1.24.0` 并列关系创建 `crosstool-build`，`crosstool-install`，`src`、`x-tools` 文件夹

```

1 | mkdir crosstool-build crosstool-install src x-tools

```

在 `crosstool-ng-1.24.0` 目录下依次执行

```
1 ./bootstrap
2 ./configure --prefix /home/liuqingshuai/Desktop/cross/crosstool-install
3 make
4 sudo make install
```

进入 `crosstool-install/bin` 目录下执行

```
1 ./ct-ng
2 ./ct-ng -v
```

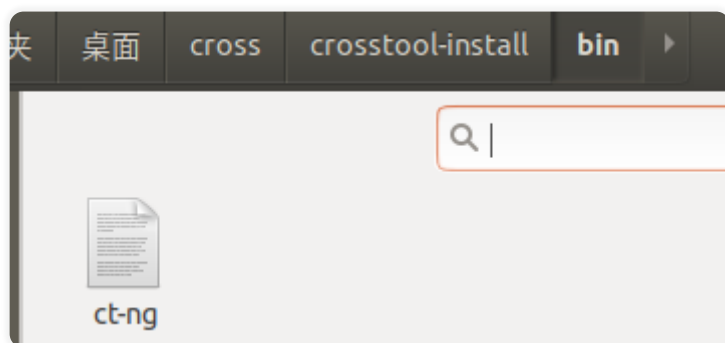
```
liuqingshuai@liuqingshuai-VirtualBox:~/桌面/cross/crosstool-install/bin$ ./ct-ng
This is crosstool-NG version 1.24.0
```

```
Copyright (C) 2008 Yann E. MORIN <yann.morin.1998@free.fr>
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
```

```
See below for a list of available actions, listed by category:
```

```
liuqingshuai@liuqingshuai-VirtualBox:~/桌面/cross/crosstool-install/bin$ ./ct-ng
-v
GNU Make 4.1
为 x86_64-pc-linux-gnu 编译
Copyright (C) 1988-2014 Free Software Foundation, Inc.
许可证: GPLv3+: GNU 通用公共许可证第 3 版或更新版本<http://gnu.org/licenses/gpl.
html>。
本软件是自由软件: 您可以自由修改和重新发布它。
在法律允许的范围内没有其他保证。
```

`crosstool-install/bin` 目录下可以看到如下文件



然后添加临时的环境变量（添加一下 `ct-ng` 的文件夹路径到 `PATH` 中，方便之后调用）

```
export PATH=$PATH:/home/liuqingshuai/Desktop/cross/crosstool-install/bin/
```

下面是网上说的 `export` 作用

指令说明: export PATH=**你要添加的地址**\$PATH

#配置完后可以通过echo \$PATH查看配置结果。

#生效方法: 立即生效

#有效期限: 临时改变, 只能在当前的终端窗口中有效, 当前窗口关闭后就会恢复原有的path配置

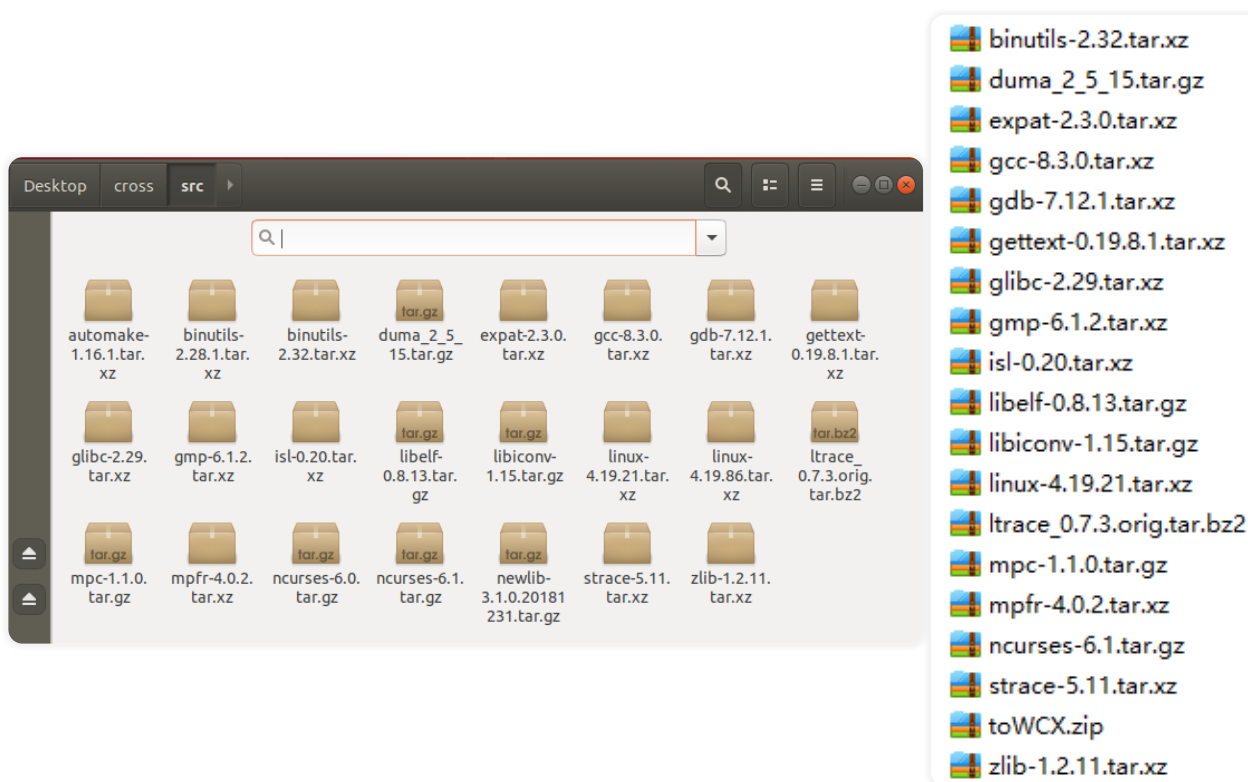
#用户局限: 仅对当前用户

也就是如果你下次打开终端窗口这个path就失效了

如果你会用vim修改, 建议用vi ~/.bashrc在最下面添加PATH (如果path是空的就别用这个方法, 先用临时变量把), 然后保存退出, 使用source ~/.bashrc使其生效, 这里不介绍这个方法了, 我们接下来复制config文件, 准备build

临时环境变量添加后, ct-ng可以在别的文件路径也能使用了

src 文件夹有如下压缩包 (谷歌搜一下, 下载), 左侧截图多了几个不需要的, 以右侧为准 (当然, 多下载肯定不会出错)



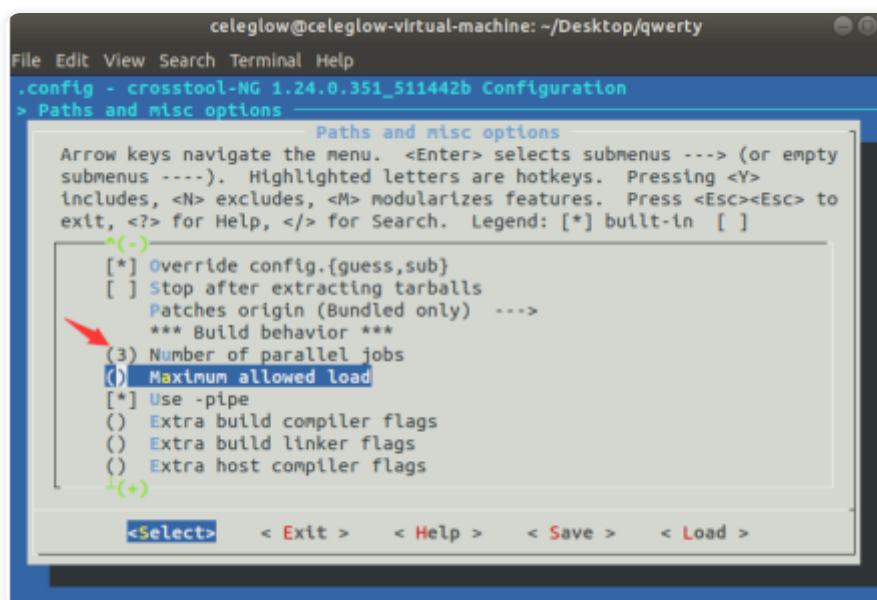
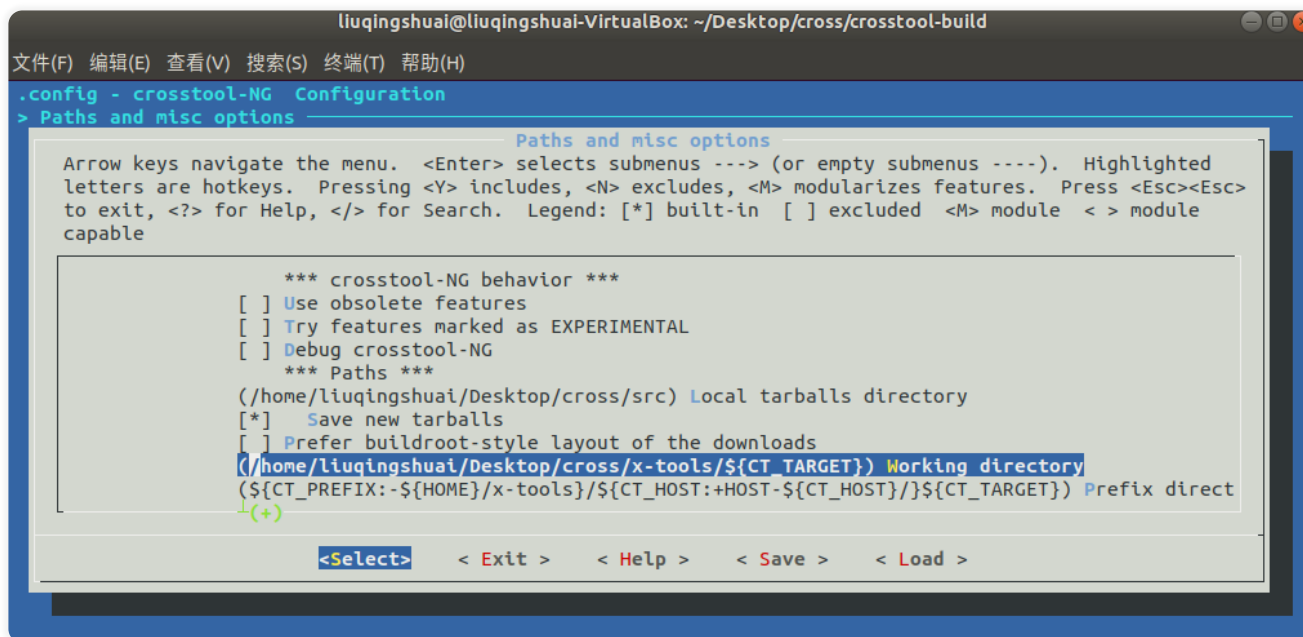
在 **crosstool-build** 目录下执行

```
1 | ct-ng menuconfig
```

然后做下述修改

选择 **Paths and misc options**, 找到 **Working directory** 和 **Local tarballs directory** 修改

还可以设置一下 Number of parallel jobs (并行任务的数量, 非必要)



选择 Target options , 架构选 arm , Architecture level 填写为 armv7-a , (你的界面应该有一个ev4, 把这个去掉才能出现 architecture level)

Target options

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 []

```
Target Architecture (arm) --->
*** Options for arm ***
Default instruction set mode (arm) --->
[ ] Use Thumb-interworking (READ HELP) (NEW)
-*- Use EABI
( ) Suffix to the arch-part
[ ] Omit vendor part of the target tuple
*** Generic target options ***
[ ] Build a multilib toolchain (READ HELP!!!)
[*] Attempt to combine libraries into a single directory
+ (+)
```

<Select>

< Exit >

< Help >

< Save >

< Load >

Target options

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 []

^(-)

```
Endianness: (Little endian) --->
Bitness: (32-bit) --->
*** Target optimisations ***
(armv7-a) Architecture level
( ) Emit assembly for CPU
( ) Tune for CPU
( ) Use specific FPU (NEW)
Floating point: (auto (let gcc decide)) --->
( ) Target CFLAGS
( ) Target LDFLAGS
```

选择 Operating System , Target OS 改为 linux , 后面的版本号不用管, 之后会手动修改

Operating System

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 []

```
Target OS (linux) --->
*** Options for linux ***
Source of linux (Released tarball) --->
Version of linux (4.20.8) --->
Kernel verbosity: (Simplified) --->
[*] Check installed headers
*** Common kernel options ***
[*] Build shared libraries
```

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

编辑 `.config` (在 `crosstool-build` 目录下, `.config` 文件通过 `ls` 看不到, 需要 `ls -a`)(上一步 `ct-ng menuconfig` 之后会自动生成 `.config` 文件)

```
1 | vim .config
```

修改 `CT_GLIBC_MIN_KERNEL` 为 `4.19.86`

```
CT_GLIBC_FORCE_UNWIND=y
# CT_GLIBC_LOCALES is not set
# CT_GLIBC_KERNEL_VERSION_NONE is not set
CT_GLIBC_KERNEL_VERSION_AS_HEADERS=y
# CT_GLIBC_KERNEL_VERSION_CHOSEN is not set
CT_GLIBC_MIN_KERNEL="4.19.86"
CT_GLIBC_SSP_DEFAULT=y
# CT_GLIBC_SSP_NO is not set
# CT_GLIBC_SSP_YES is not set
# CT_GLIBC_SSP_ALL is not set
# CT_GLIBC_SSP_STRONG is not set
# CT_GLIBC_ENABLE_WERROR is not set
CT_ALL_LIBC_CHOICES="AVR_LIBC BIONIC GLIBC MINGW_W64 MOXIEBOX MUSL NEWLIB NONE UCLIBC"
CT_LIBC_SUPPORT_THREADS_ANY=y
CT_LIBC_SUPPORT_THREADS_NATIVE=y
#
-- 插入 --
```

438,29

58%

`CT_LINUX_VERSION` 为 `4.19.86`


```

# CT_LINUX_V_3_10 is not set
# CT_LINUX_V_3_4 is not set
# CT_LINUX_V_3_2 is not set
# CT_LINUX_NO_VERSIONS is not set
CT_LINUX_VERSION="4.19.86"
CT_LINUX_MIRRORS="$(CT_Mirrors kernel.org linux ${CT_LINUX_VERSION})"
CT_LINUX_ARCHIVE_FILENAME="@{pkg_name}-${version}"
CT_LINUX_ARCHIVE_DIRNAME="@{pkg_name}-${version}"
CT_LINUX_ARCHIVE_FORMATS=".tar.xz .tar.gz"
CT_LINUX_SIGNATURE_FORMAT="unpacked/.sign"
CT_LINUX_later_than_4_8=y
CT_LINUX_4_8_or_later=y
CT_LINUX_later_than_3_7=y
CT_LINUX_3_7_or_later=y
CT_LINUX_later_than_3_2=y

```

在 `crosstool-build` 下执行下面指令（时间很长）

```
1 | ct-ng build
```

```

liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/crosstool-build$ ct-ng build
[INFO ] Performing some trivial sanity checks
[WARN ] Number of open files 1024 may not be sufficient to build the toolchain;
        increasing to 2048
[INFO ] Build started 20220604.220314
[INFO ] Building environment variables
[EXTRA] Preparing working directories

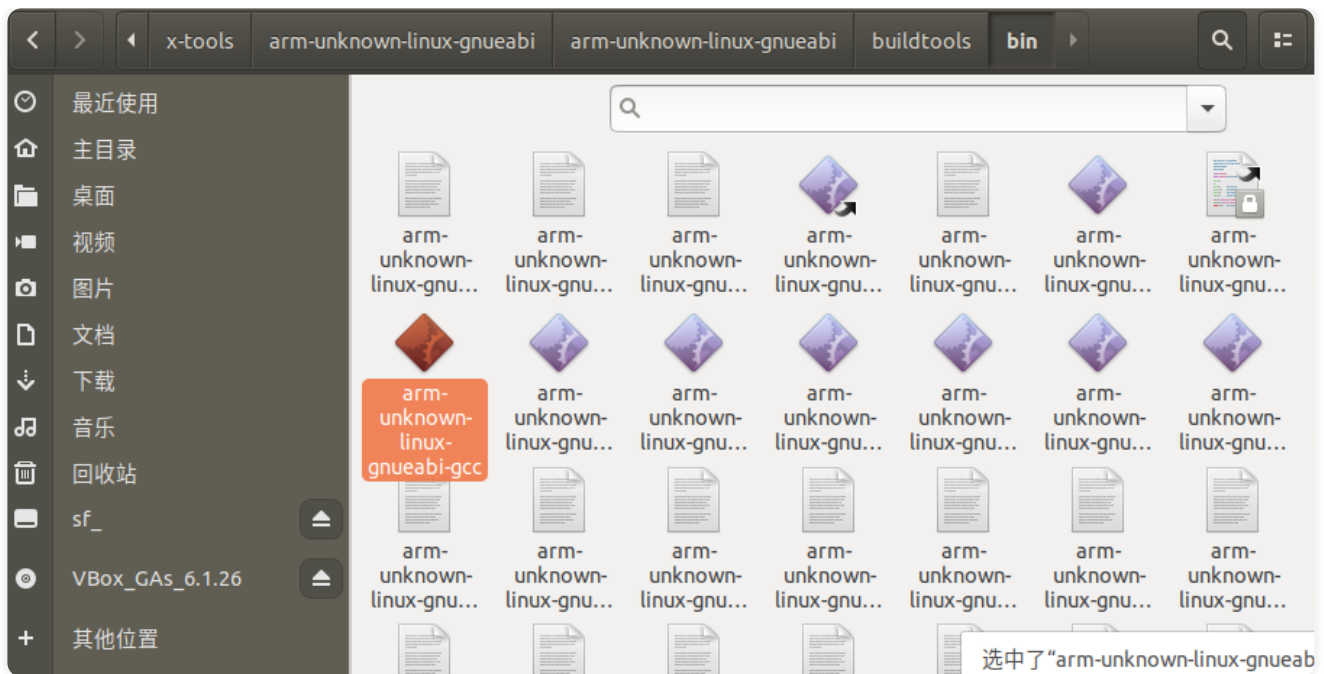
```

```

[EXTRA]      '--> lib (gcc)  lib (os)
[INFO ] Installing final gcc compiler: done in 472.20s (at 68:29)
[INFO ] =====
[INFO ] Finalizing the toolchain's directory
[INFO ] Stripping all toolchain executables
[EXTRA] Installing the populate helper
[EXTRA] Installing a cross-ldd helper
[EXTRA] Creating toolchain aliases
[EXTRA] Removing installed documentation
[EXTRA] Collect license information from: /home/liuqingshuai/Desktop/cross/x-tools/alphaev4-unknown-linux-gnu/alphaev4-unknown-linux-gnu/src
[EXTRA] Put the license information to: /home/liuqingshuai/x-tools/alphaev4-unknown-linux-gnu/share/licenses
[INFO ] Finalizing the toolchain's directory: done in 5.78s (at 68:34)
[INFO ] Build completed at 20220606.111840
[INFO ] (elapsed: 68:33.21)
[INFO ] Finishing installation (may take a few seconds)...
[68:34] / liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/crosstool-build$

```

生成的 `gcc` 在如下目录 `/home/liuqingshuai/Desktop/cross/x-tools/arm-unknown-linux-gnueabi/arm-unknown-linux-gnueabi/buildtools/bin`



测试一下这个 `gcc` 的版本(在上面那个 `bin` 目录下测试)

```
1 | ./arm-unknown-linux-gnueabi-gcc --version
```

```
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/cross/x-tools/arm-unknown-linux-gnueabi/arm-unknown-linux-gnueabi/buildtools/bin$ ./arm-unknown-linux-gnueabi-gcc --version
arm-unknown-linux-gnueabi-gcc (crosstool-NG 1.24.0) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

`gcc` 的绝对位置如下 `/home/liuqingshuai/Desktop/cross/x-tools/arm-unknown-linux-gnueabi/arm-unknown-linux-gnueabi/buildtools/bin/arm-unknown-linux-gnueabi-gcc`

`arm` 目录新建一个 `helloworld.c` 然后编译



```
1 | /home/liuqingshuai/Desktop/cross/x-tools/arm-unknown-linux-gnueabi/arm-unknown-linux-gnueabi/buildtools/bin/arm-unknown-linux-gnueabi-gcc -static hello.c -o hello
```

```
liuqingshuai@liuqingshuai-VirtualBox:~/Desktop/arm$ /home/liuqingshuai/Desktop/cross/x-tools/arm-unknown-linux-gnueabi/arm-unknown-linux-gnueabi/buildtools/bin/arm-unknown-linux-gnueabi-gcc -static hello.c -o hello
```

采用实验 1 的挂载

```
1 | sudo mount -o loop ./rootfs.ext3 /tmp
```

然后把这个 `world` 可执行文件 copy 到 `tmp` 目录里，如果遇到权限不够的错误，则在 `arm` 目录下执行指令，然后再复制（会出现读一行什么玩意的error，不用管），权限就够了

```
1 | sudo nautilus
```

然后取消挂载

```
1 | sudo umount /tmp
```

`arm` 目录下启动 `arm` 模拟器

```
1 | qemu-system-arm -M vexpress-a9 -kernel ./zImage -nographic -m 512M -smp 4 -sd  
./rootfs.ext3 -dtb vexpress-v2p-ca9.dtb -append "init=/linuxrc root=/dev/mmcblk0  
rw rootwait earlyprintk console=ttyAMA0"
```

然后运行可执行文件，成功输出

```
/bin/sh: can't access tty; job control turned off
/ # ls
bin
dev
etc
hello
lib
linuxrc
lost+found
proc
sbin
sys
systemd-private-24ce8b274c5e41bf8b6e07c8bb0a493f-systemd-hostnamed.service-Ma9Ss
1
systemd-private-24ce8b274c5e41bf8b6e07c8bb0a493f-systemd-hostnamed.service-YV4E3
e
systemd-private-24ce8b274c5e41bf8b6e07c8bb0a493f-systemd-hostnamed.service-bAAHt
D
tmp
usr
whatfuck
/ # ./hello
lqs say to you:hello
```

至此，实验完了

H3 2.3实验结果

成功生成了 `arm` 的 `gcc` 编译工具，并成功生成 `arm` 平台下可执行文件，同时在 `qemu` 模拟器里成功执行

H2 三、应用程序开发

H3 3.1实验目的

利用实验一编译出的操作系统和实验二构建的编译工具链，完成一个基于c语言的SHA-1应用程序开发。编译产生的可执行文件能在 `qemu` 中执行。

H3 3.2算法步骤

对于任意长度的明文，SHA1首先对其进行分组，使得每一组的长度为512位，然后对这些明文分组反复重复处理。

对于每个明文分组的摘要生成过程如下：

- (1) 将512位的明文分组划分为16个子明文分组，每个子明文分组为32位。
- (2) 申请5个32位的链接变量，记为A、B、C、D、E。
- (3) 16份子明文分组扩展为80份。
- (4) 80份子明文分组进行4轮运算。

- (5) 链接变量与初始链接变量进行求和运算。
- (6) 链接变量作为下一个明文分组的输入重复进行以上操作。
- (7) 最后，5个链接变量里面的数据就是SHA1摘要。

📄 sha1.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define SHA1_ROTL(a,b) (SHA1_tmp=(a),((SHA1_tmp>>(32-b))&(0x7fffffff>>(31-b)))|
   (SHA1_tmp<<b))
4  #define SHA1_F(B,C,D,t) ((t<40)?((t<20)?((B&C)|((~B)&D)):(B^C^D)):(t<60)?
   ((B&C)|(B&D)|(C&D)):(B^C^D)))
5  long SHA1_tmp;
6  char* StrSHA1(const char* str, long long length, char* sha1){
7      /*
8       计算字符串SHA-1
9       参数说明:
10      str          字符串指针
11      length       字符串长度
12      sha1         用于保存SHA-1的字符串指针
13      返回值为参数sha1
14      */
15      char *pp, *ppend;
16      long l, i, K[80], W[80], TEMP, A, B, C, D, E, H0, H1, H2, H3, H4; //K和W是32位
   的数组
17      H0 = 0x67452301, H1 = 0xEFCDAB89, H2 = 0x98BADCFE, H3 = 0x10325476, H4 =
   0xC3D2E1F0; //初始化变量
18      for (i = 0; i < 20; K[i++] = 0x5A827999);
19      for (i = 20; i < 40; K[i++] = 0x6ED9EBA1);
20      for (i = 40; i < 60; K[i++] = 0x8F1BBCDC);
21      for (i = 60; i < 80; K[i++] = 0xCA62C1D6);
22      l = length + ((length % 64 > 56) ? (128 - length % 64) : (64 - length %
   64)); //l=64
23      printf("l=%d\n", l);
24      if (!(pp = (char*)malloc((unsigned long)l))) return 0;
25      for (i = 0; i < length; pp[i + 3 - 2 * (i % 4)] = str[i], i++); //为llehw
   ,odlro
26      for (pp[i + 3 - 2 * (i % 4)] = 128, i++; i < l; pp[i + 3 - 2 * (i % 4)] =
   0, i++);
27      *((long*)(pp + l - 4)) = length << 3;
28      *((long*)(pp + l - 8)) = length >> 29;
29      for (ppend = pp + l; pp < ppend; pp += 64){
30          for (i = 0; i < 16; W[i] = ((long*)pp)[i], i++);
31          for (i = 16; i < 80; W[i] = SHA1_ROTL((W[i - 3] ^ W[i - 8] ^ W[i - 14] ^
   W[i - 16]), 1), i++);
32          A = H0, B = H1, C = H2, D = H3, E = H4;
```

```

33     for (i = 0; i < 80; i++){
34         TEMP = SHA1_ROTL(A, 5) + SHA1_F(B, C, D, i) + E + W[i] + K[i];
35         E = D, D = C, C = SHA1_ROTL(B, 30), B = A, A = TEMP;
36     }
37     H0 += A, H1 += B, H2 += C, H3 += D, H4 += E;
38 }
39 free(pp - 1);
40 sprintf(shal, "%08X%08X%08X%08X%08X", H0, H1, H2, H3, H4);
41 return shal;
42 }
43
44 int main(){
45     printf("你好\n");
46     char shal[41]={0}; //shal用于保存计算结果
47     StrSHA1("hello, world", 12, shal); //计算字符串"hello, world"前12位的shal
48     printf(shal);
49     printf("\n");
50 }

```

终端执行

```

1  /home/liuqingshuai/Desktop/cross/x-tools/arm-unknown-linux-gnueabi/arm-unknown-
   linux-gnueabi/buildtools/bin/arm-unknown-linux-gnueabi-gcc -static sha1.c -o
   sha1Arm
2  sudo mount -o loop ./rootfs.ext3 /tmp
3  sudo nautilus
4  拷贝 sha1Arm 到 tmp 目录
5  sudo umount /tmp
6  qemu-system-arm -M vexpress-a9 -kernel ./zImage -nographic -m 512M -smp 4 -sd
   ./rootfs.ext3 -dtb vexpress-v2p-ca9.dtb -append "init=/linuxrc root=/dev/mmcblk0
   rw rootwait earlyprintk console=ttyAMA0"

```

```
Please press Enter to activate this console.
/bin/sh: can't access tty; job control turned off
/ # ls
bin
dev
etc
hello
lib
linuxrc
lost+found
proc
sbin
sha1Arm
sys
systemd-private-24ce8b274c5e41bf8b6e07c8bb0a493f-systemd-hostnamed.service-Ma9Ss
1
systemd-private-24ce8b274c5e41bf8b6e07c8bb0a493f-systemd-hostnamed.service-YV4E3
e
systemd-private-24ce8b274c5e41bf8b6e07c8bb0a493f-systemd-hostnamed.service-bAAHt
D
systemd-private-da2fee322acc4f4698419f83df0b6907-systemd-hostnamed.service-SKl8X
R
tmp
usr
whatfuck
/ # ./sha1Arm
你好
l=64
B7E23EC29AF22B0B4E41DA31E868D57226121C84
/ #
```

在网站上验证 `hello, world` (中间有一个空格)的sha1加密结果, 发现完全一样。

SHA-1在线加密工具

hello, world

加密 ☒ 大写字母

B7E23EC29AF22B0B4E41DA31E868D57226121C84

SHA (Secure Hash Algorithm) 中文翻译为“安全散列算法”, 是美国国家安全局 (NSA) 设计, 美国国家标准与技术研究院 (NIST) 发布的一系列密码散列函数。正式名称为 SHA 的家族第一个成员发布于1993年。然而现在的人们给它取了一个非正式的名称 SHA-0 以避免与它的后继者混淆。两年之后, 第一个 SHA 的后继者 SHA-1 发布了。另外还有四种变体, 曾经发布以提升输出的范围和变更一些细微设计: SHA-224, SHA-256, SHA-384 和 SHA-512 (这些有时候也被称做 [SHA-2](#))。

SHA-1 散列函数加密算法输出的散列值为40位十六进制数字串, 可用于验证信息的一致性, 防止被篡改。本页面的 SHA-1 在线加密工具可对字符串进行 SHA-1 加密, 并可转换散列值中字母的大小写。

QQXIUZI.CN 千千秀字 用户反馈

H3 3.3实验结果

成功写了一个能计算 `SHA-1` 的C语言程序, 并利用自己的交叉编译工具链生成了可执行文件, 在 `qemu` 模拟器里成功执行

