实验一：内核配置

1.安装交叉编译器：

使用以下代码下载解压交叉编译器

git clone https://github.com/friendlyarm/prebuilts.git -b master --depth 1

cd prebuilts/gcc-x64

cat toolchain-4.9.3-armhf.tar.gz* | sudo tar xz -C /

然后设置环境变量：

vi /etc/profile

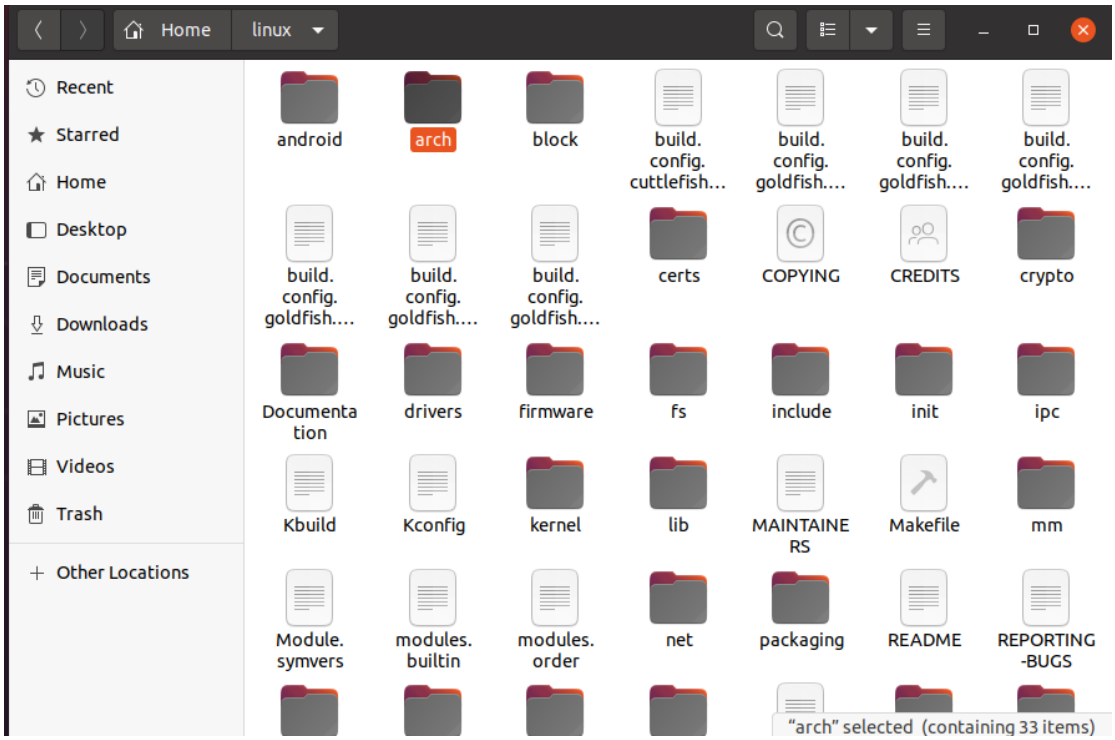
```
export PATH=/opt/FriendlyARM/toolchain/4.9.3/bin:$PATH
export GCC_COLORS=auto
```

source /etc/profile 执行环境变量

使用 arm-linux-gcc -v 查看交叉编译器是否安装成功

```
root@ubuntu: /home/lgx
root@ubuntu:/home/lgx# vi /etc/profile
root@ubuntu:/home/lgx# arm-linux-gcc -v
使用内建 specs。
COLLECT_GCC=arm-linux-gcc
COLLECT_LTO_WRAPPER=/opt/FriendlyARM/toolchain/4.9.3/libexec/gcc/arm-cortexa9-l
inux-gnueabihf/4.9.3/lto-wrapper
目标：arm-cortexa9-linux-gnueabihf
配置为：/work/toolchain/build/src/gcc-4.9.3/configure --build=x86_64-build_pc-l
inux-gnu --host=x86_64-build_pc-linux-gnu --target=arm-cortexa9-linux-gnueabihf
 --prefix=/opt/FriendlyARM/toolchain/4.9.3 --with-sysroot=/opt/FriendlyARM/tool
chain/4.9.3/arm-cortexa9-linux-gnueabihf/sys-root --enable-languages=c,c++ --wi
th-arch=armv7-a --with-tune=cortex-a9 --with-fpu=vfpv3 --with-float=hard --with
-pkgversion=ctng-1.21.0-229g-FA --with-bugurl=http://www.friendlyarm.com/ --ena
ble-__cxa_atexit --disable-libmudflap --disable-libgomp --disable-libssp --disa
ble-libquadmath --disable-libquadmath-support --disable-libsanitizer --with-gmp
=/work/toolchain/build/arm-cortexa9-linux-gnueabihf/buildtools --with-mpfr=/wor
k/toolchain/build/arm-cortexa9-linux-gnueabihf/buildtools --with-mpc=/work/tool
chain/build/arm-cortexa9-linux-gnueabihf/buildtools --with-isl=/work/toolchain/
build/arm-cortexa9-linux-gnueabihf/buildtools --with-cloog=/work/toolchain/buil
d/arm-cortexa9-linux-gnueabihf/buildtools --with-libelf=/work/toolchain/build/a
rm-cortexa9-linux-gnueabihf/buildtools --enable-lto --with-host-libstdcxx='-sta
tic-libgcc -Wl,-Bstatic,-lstdc++,-Bdynamic -lm' --enable-threads=posix --enable
-linker-build-id --with-linker-hash-style=gnu --enable-plugin --enable-gold --d
isable-multilib --with-local-prefix=/opt/FriendlyARM/toolchain/4.9.3/arm-cortex
a9-linux-gnueabihf/sys-root --enable-long-long
线程模型：posix
gcc 版本 4.9.3 (ctng-1.21.0-229g-FA)
root@ubuntu:/home/lgx#
```

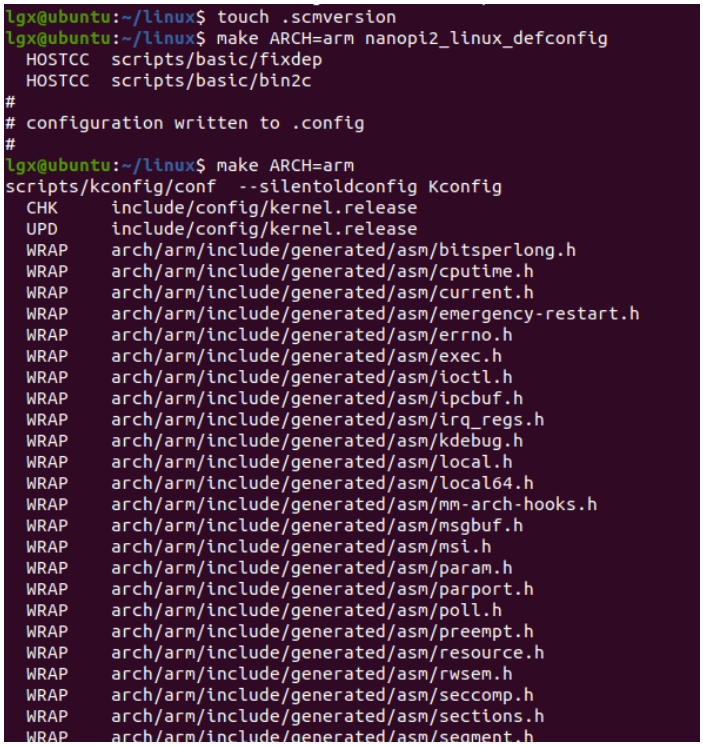
2.下载 Linux 内核：



使用 make menuconfig 对内核设置进行修改，直接保存，得到.config 文件

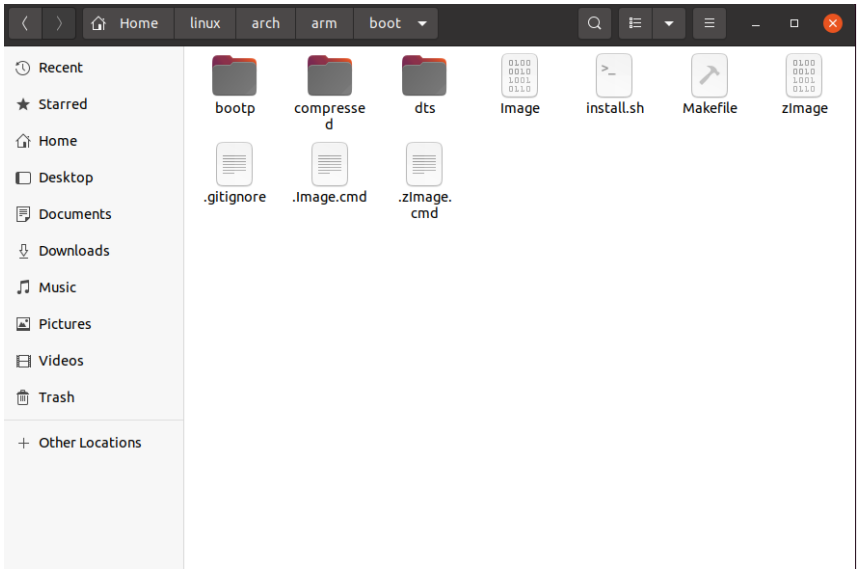
3.编译内核：

touch .scmversion
make ARCH=arm nanopi2_linux_defconfig
make ARCH=arm



```
lgx@ubuntu:~/linux$ touch .scmversion
lgx@ubuntu:~/linux$ make ARCH=arm nanopi2_linux_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/basic/bin2c
#
# configuration written to .config
#
lgx@ubuntu:~/linux$ make ARCH=arm
scripts/kconfig/conf  --silentoldconfig Kconfig
  CHK     include/config/kernel.release
  UPD     include/config/kernel.release
  WRAP    arch/arm/include/generated/asm/bitsperlong.h
  WRAP    arch/arm/include/generated/asm/cputime.h
  WRAP    arch/arm/include/generated/asm/current.h
  WRAP    arch/arm/include/generated/asm/emergency-restart.h
  WRAP    arch/arm/include/generated/asm/errno.h
  WRAP    arch/arm/include/generated/asm/exec.h
  WRAP    arch/arm/include/generated/asm/ioctl.h
  WRAP    arch/arm/include/generated/asm/ipcbuf.h
  WRAP    arch/arm/include/generated/asm/irq_regs.h
  WRAP    arch/arm/include/generated/asm/kdebug.h
  WRAP    arch/arm/include/generated/asm/local.h
  WRAP    arch/arm/include/generated/asm/local64.h
  WRAP    arch/arm/include/generated/asm/mm-arch-hooks.h
  WRAP    arch/arm/include/generated/asm/msgbuf.h
  WRAP    arch/arm/include/generated/asm/msi.h
  WRAP    arch/arm/include/generated/asm/param.h
  WRAP    arch/arm/include/generated/asm/parport.h
  WRAP    arch/arm/include/generated/asm/poll.h
  WRAP    arch/arm/include/generated/asm/preempt.h
  WRAP    arch/arm/include/generated/asm/resource.h
  WRAP    arch/arm/include/generated/asm/rwsem.h
  WRAP    arch/arm/include/generated/asm/seccomp.h
  WRAP    arch/arm/include/generated/asm/sections.h
  WRAP    arch/arm/include/generated/asm/segment.h
```

编译完成后，在 arch/arm/boot 中，可以看到得到了 zImage 文件



4.添加外部驱动，先编写 Makefile 文件：

obj-$(CONFIG_DRIVER_VMALLOC) += driver_kernel.o

5.然后编写 Kconfig 文件：

参考源码中有错误：

```
#
# DRIVER test subsystem configuration
#
menu "DRIVER  KMALLOC support"
    config DRIVER_VMALLOC
    tristate "Driver_test is supported"
#   ---help---
#   Driver_test use vmalloc .

endmenu
```

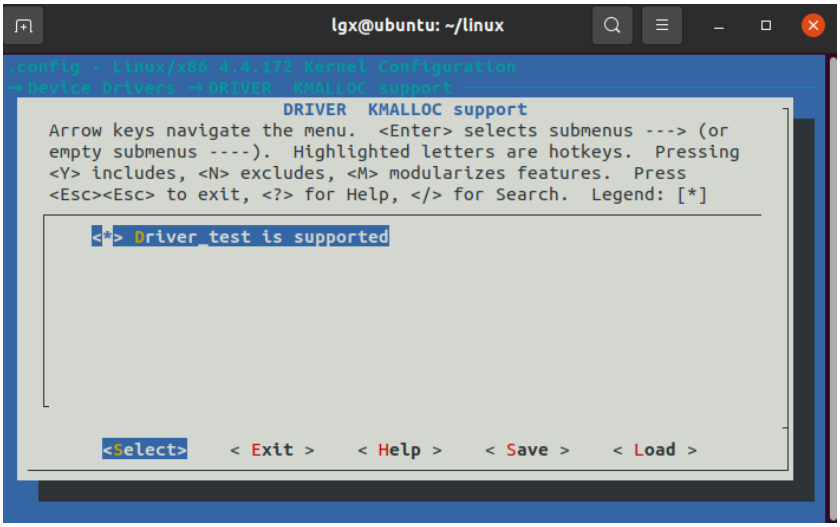
---help--- 和 Driver_test use vmalloc 这两行应该被注释掉，否则编译内核时会报错

6.再对 drivers 文件夹中的 Kconfig 和 Makefile 文件进行修改

Kconfig 中添加 source “drivers/drivertest/Kconfig”

Makefile 中添加 obj-$(CONFIG_DRIVER_VMALLOC) += drivertest/

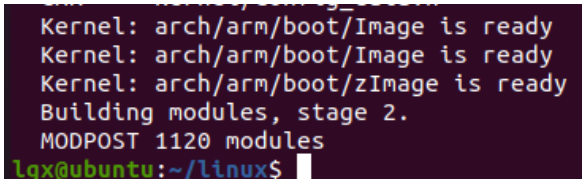
7.完成以上所有配置后，使用 make menuconfig 重新配置内核，加载新驱动 myalloc：



选择*表示选择该驱动，然后选择 Save 保留修改

8.完成内核设置后，再次使用以下命令对内核进行编译

touch .scmversion
make ARCH=arm nanopi2_linux_defconfig
make ARCH=arm



得到了映像文件 zImage