

5.2 U盘启动

5.2.1 制作启动U盘

我们准备一个U盘（最小不要小于4G，最大不要大于32G），U盘有且只有一个分区，U盘格式化成FAT32分区，如不满足要求，请格式化您的U盘。

U盘准备好以后，在U盘根目录下建立“boot”文件夹，将资料中的带文件系统的镜像“vmlinuz”到U盘的boot文件夹下面。

拷贝LS2K0300开发板资料\05_文件系统镜像目录下的任意文件系统“rootfs.cpio.gz”压缩文件到U盘根目录的“boot”文件夹下，接下来我们在U盘的“boot”文件夹下建立“boot.cfg”启动配置文件，并在“boot.cfg”文件里面输入下面的内容：

```
timeout 5
default 0
showmenu 1
title kernel or fs on usb
    kernel /dev/fs/fat@usb0/boot/vmlinuz
    initrd /dev/fs/fat@usb0/boot/rootfs.cpio.gz
    args console=tty console=ttyS0,115200
```

代码解释：

boot.cfg 为系统启动配置文件。

timeout 5 //等待5 秒

default 0 //默认引导第0 个title

title kernel or fs on usb //启动的title，显示到PMON界面，方便分辨启动的内核、文件系统以及启动参数

kernel /dev/fs/fat@usb0/boot/vmlinuz //内核路径

initrd /dev/fs/fat@usb0/boot/rootfs.cpio.gz //文件系统路径

args console=tty console=ttyS0,115200 //启动参数，设置打印终端为ttyS0，并在屏幕上显示打印信息，波特率为115200，后面还可以跟上其他参数，用来指引内核启动后挂载的文件系统、log级别等。

写好系统启动配置文件后，保存并退出U盘。这样我们的启动U盘就做好了。

5.3.3 设置网络

使用网线将开发板连接到路由器上，在pmon启动的过程中，我们一直按着PC键盘的“c”按键，使pmon进入命令行模式使用命令“ifaddr syn1 192.168.1.10”设置开发板的IP地址和虚拟机Ubuntu的IP在同一个网段下，并可以ping通虚拟机，如下图所示：

5.3.4 加载PMON镜像

注意：加载PMON镜像相当于重新烧写PMON镜像，需要确定PMON镜像是没有问题的，否则加载了错误的PMON镜像会导致板子不能正常启动，而没有Ejtag仿真器会导致板子永远起不来。所以不建议在没有Ejtag的情况下更新PMON。

网络设置完成后，输入命令`load -f 0xbfc00000 -r tftp://192.168.1.177/gzrom-dtb.bin`，其中192.168.1.177为虚拟机Ubuntu 的IP 地址，根据实际情况修改为对应的IP地址。

5.3.5 加载内核镜像

在PMON 目录下输入命令`load tftp://192.168.1.177/vmlinuz` 加载内核到开发板，其中192.168.1.177 为虚拟机Ubuntu 的IP 地址。

这里加载的只是内核文件，内核启动后需要挂在文件系统，因此需要烧写完文件系统以后才可以使用g 命令启动开发板。

注意，这里只是加载到了内存，并没有烧写到flash里面，断电后加载的镜像会丢失。

5.3.6 加载文件系统

在 PMON 目录下输入命令 `initrd tftp://192.168.1.177/rootfs.cpio.gz` 加载文件系统 (buildroot 根文件系统)，其中192.168.1.177 为虚拟机Ubuntu的IP 地址。

加载完成后使用命令`g console=tty console=ttyS0,115200` 启动系统。

启动成功如下图所示：

注意，这里只是加载到了内存，并没有烧写到flash 里面。

5.4.1 加载内核镜像

加载内核镜像有两种方式，第一种是通过网络的方式进行加载，第二种是通过U盘进行加载。

5.4.1.1 网络加载内核镜像

网络加载需要确定板子可以正常联网，虚拟机安装了tftp服务。然后启动板子到PMON命令行停下，如下图所示：

设置网口IP地址，使用命令**ifaddr syn0 192.168.1.56**

具体的IP需要根据实际的局域网进行修改，然后使用**set**命令查看IP，如下图所示：

```

PMON> ifaddr syn0 192.168.1.56
bootp=f00b740
synopGMAC_linux_open called
Version = 0x1137
MacAddr = 0x0    0x55    0x7b    0xb5    0x7d    0xf7

===phy HALFDUPLEX MODE
DMA status reg = 0x0 before cleared!
DMA status reg = 0x0 after cleared!
register poll interrupt: gmac 0
==arp_ifinit done
PMON> set
    FR = 1
    al = (usb0,0)/boot/vmlinuz
    append = "'console=ttyS0,115200 loglevel=7 root=/dev/sda'"
    all = /dev/mtd0
    ethaddr = 00:00:00:00:00:00
    memsize = 176
    highmemsize = 768
    cpuclock = 500000000
    vramsize = 16
    sharevram = 0
    systype = loongson
    brkcmd = "l -r @cpc 1"
    datasize = -b          [-b -h -w -d]
    dlecho = off           [off on lfeed]
    dlproto = none         [none XonXoff EtxAck]
    bootp = no             [no sec pri save]
    hostport = tty0
    inalpha = hex          [hex symbol]
    inbase = 16             [auto 8 10 16]
    moresz = 10
    prompt = "PMON> "
    regstyle = sw           [hw sw]
    regsize = 32            [32 64]
    rptcmd = trace         [off on trace]
    trabort = ^K
    ulcr = cr              [cr lf crlf]
    uleof = %
    showsym = yes          [no yes]
    ffmt = both            [both double single none]
    fpdis = yes            [no yes]
    mtdparts = nand-flash:30M@0(kernel)ro,-(rootfs)
    bootdelay = 3
    syn0.ipaddr = 192.168.1.56
PMON>

```

将内核文件放到虚拟机下tftp的根目录下，使用命令 **devcp tftp://192.168.1.209/vmlinuz /dev/mtd0**，即可将内核文件通过网络拷贝到NAND Flash上的第一个分区中。

5.4.1.2 U盘加载内核镜像

通过U盘拷贝内核文件需要确定U盘格式为FAT32，然后将内核文件放到U盘上，开机进入到PMON命令行，输入命令 **devcp /dev/fs/fat@usb0/vmlinuz /dev/mtd0**，即可将内核文件通过U盘加载到NAND Flash上。

5.4.2 拷贝文件系统

拷贝文件系统和加载内核镜像相似，只是将对应的内核文件名更换为文件系统的名字即可。

5.4.3 从NAND上启动

PMON中有set命令设置环境变量，可以设置默认从NAND上加载。设置环境变量依次执行以下命令：

```
set al /dev/mtd0
set append "console=ttyS0,115200 init=/linuxrc rootfstype=yaffs2 rw
root=/dev/mtdblock1"
```

5.5 SDIO NAND启动

LS2K0500开发板在核心板上板载一个1G大小的SDIO接口的NAND，我们可以挂载这个NAND进行启动。

使用SDIO NAND需要确定加载内核的方式，可以选择U盘加载内核或者是NAND Flash加载内核。使用NAND Flash加载内核会导致LCD的触摸、CAN接口不能使用。

5.5.1 文件系统解压

在buildroot编译好文件系统镜像后，需要确定生成一个tar的压缩包，这个压缩包是我们要用到的。网盘中也提供了这个镜像，位置在：

然后确定U盘是在5.2节中可以启动的U盘，然后我们将tar包也放在这个U盘中，通过U盘启动开发板进入到文件系统中。如下图所示：

```
[ 34.015793] Run /init as init process
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting mdev... OK
modprobe: can't change directory to '/lib/modules': No such file or directory
Saving random seed: [ 35.537117] random: dd: uninitialized urandom read (512 bytes read)
OK
Starting system message bus: [ 35.587913] random: dbus-uuidgen: uninitialized urandom re
[ 35.616486] random: dbus-uuidgen: uninitialized urandom read (8 bytes read)
done
Starting network: [ 35.743900] Generic PHY stmmac-1:00: attached PHY driver [Generic PHY]
[ 35.879622] stmmaceth 1f030000.ethernet: Failed to reset the dma
[ 35.908058] stmmaceth 1f030000.ethernet eth1: stmmac_hw_setup: DMA engine initializatio
[ 35.939673] stmmaceth 1f030000.ethernet eth1: stmmac_open: Hw setup failed
RTNETLINK answers: Device or resource busy
FAIL
Starting DHCP server: FAIL

Welcome to Buildroot
buildroot login: root
#
```

使用命令 **ls /dev/** 可以看到以下几个节点，其中 **mmcblk0** 和 **mmcblk0p1** 是 SDIO NAND 生成的节点，**sda** 和 **sda1** 是 U 盘生成的节点，如下图所示：

```
ls /dev/
kmsg          tt
log           tt
mem           tt
memory_bandwidth tt
mmcblk0       tt
mmcblk0p1     tt
mpt2ctl       tt
mpt3ctl       tt
mqueue        tt
network_latency tt
network_throughput tt
null          tt
port          tt
psaux         tt
ptmx          tt
pts           tt
random        tt
sda           tt
sda1          tt
shm           tt
snapshot      tt
snd           tt
stderr        tt
```

然后我们格式化掉 SDIO NAND 以便于解压文件系统：

```
mkfs.ext4 /dev/mmcblk0
```

然后我们创建两个目录用来分别挂载 SDIO NAND 和 U 盘。

```
mkdir /mnt/udisk
```

```
mkdir /mnt/mmc
```

然后将 U 盘和 SDIO NAND 分别挂载上去：

```
mount /dev/sda1 /mnt/udisk/
```

```
mount /dev/mmcblk0 /mnt/mmc/
```

解压文件系统：

```
tar -vxf /mnt/udisk/boot/rootfs.tar -C /mnt/mmc/ && sync
```

等待解压完成如下图所示：

```
./var/lib/
./var/lib/alsa/
./var/lib/arpd/
./var/lib/dbus
./var/lib/dhcp
./var/lib/misc
./var/lib/sudo/
./var/lib/sudo/lectured/
./var/lock
./var/log
./var/run
./var/spool
./var/tmp
./var/www/
#
```

然后将 SDIO NAND 和 U 盘进行解挂。


```
umount /dev/sda1  
umount /mnt/mmc/
```

5.5.2 使用U盘加载内核

使用U盘加载内核需要按照5.2节制作启动U盘，然后boot.cfg文件输入以下内容：

```
timeout 5  
default 0  
showmenu 1  
  
title kernel or fs on usb  
kernel /dev/fs/fat@usb0/boot/vmlinuz  
args console=tty console=ttyS0,115200 loglevel=7 root=/dev/mmcblk0
```

然后将U盘插到板子的usb接口上启动即可。

5.5.3 使用NAND Flash加载内核

使用NAND Flash加载内核需要将内核文件按照5.4节拷贝到NAND Flash中，然后进入到PMON下设置环境变量：

```
set al /dev/mtd0  
set append "'console=ttyS0,115200 loglevel=7 root=/dev/mmcblk0'"
```

然后重启板子即可。

5.5.4 启动系统

上面的设置好后，重启板子，PMON启动以后会自动加载内核，内核启动以后会自动挂载SDIO NAND。

第六章 LS2K0300开发板系统编译

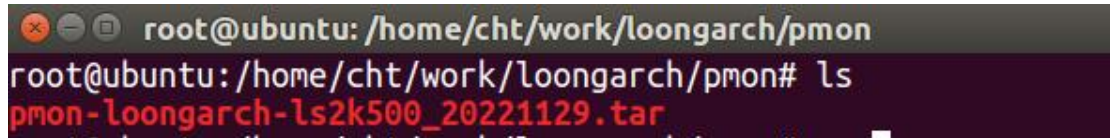
这一章我们介绍安装LS2K0300开发板的交叉编译工具链以及编译PMON、Linux内核、buildroot文件系统。

6.1 安装交叉编译工具链

将网盘资料中的交叉编译工具链拷贝到Ubuntu下解压，使用命令“tar -xvf loongarch64-linux-gnu-2022-01-26-vector.tar.gz -C /opt”将工具链解压到/opt目录下。因为编译脚本中已经将交叉编译工具链的路径配置好且为绝对路径，因此最好不要将交叉编译工具链解压到其他位置。

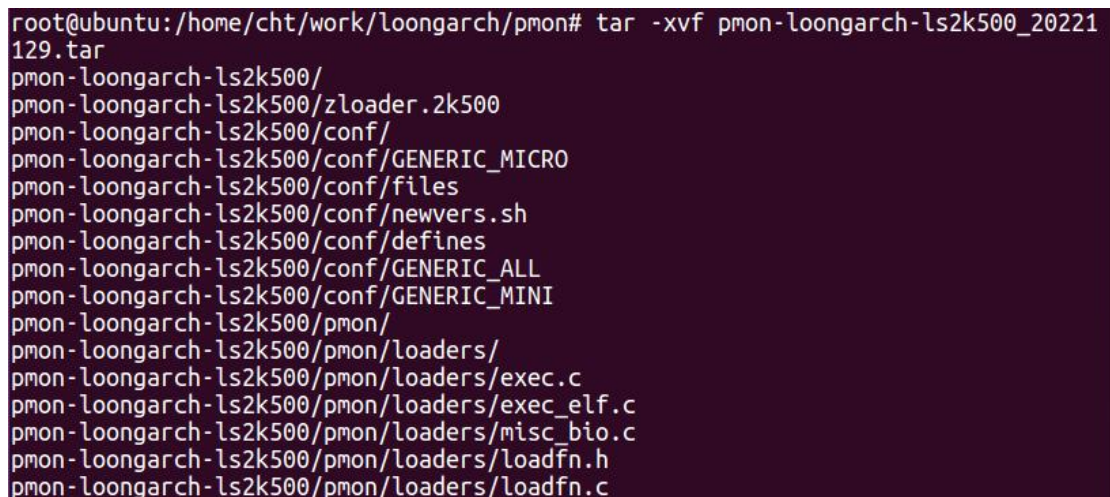
6.2 编译PMON

将PMON的源码拷贝源码到Ubuntu的目录中,如下图所示:



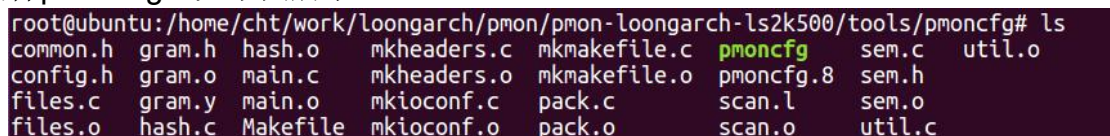
```
root@ubuntu: /home/cht/work/loongarch/pmon
root@ubuntu: /home/cht/work/loongarch/pmon# ls
pmon-loongarch-ls2k500_20221129.tar
```

输入“tar -xvf pmon-loongarch-ls2k500_20221129.tar”对PMON源码进行解压如下图所示:



```
root@ubuntu: /home/cht/work/loongarch/pmon# tar -xvf pmon-loongarch-ls2k500_20221129.tar
pmon-loongarch-ls2k500/
pmon-loongarch-ls2k500/zloader.2k500
pmon-loongarch-ls2k500/conf/
pmon-loongarch-ls2k500/conf/GENERIC_MICRO
pmon-loongarch-ls2k500/conf/files
pmon-loongarch-ls2k500/conf/newvers.sh
pmon-loongarch-ls2k500/conf/defines
pmon-loongarch-ls2k500/conf/GENERIC_ALL
pmon-loongarch-ls2k500/conf/GENERIC_MINI
pmon-loongarch-ls2k500/pmon/
pmon-loongarch-ls2k500/pmon/loaders/
pmon-loongarch-ls2k500/pmon/loaders/exec.c
pmon-loongarch-ls2k500/pmon/loaders/exec_elf.c
pmon-loongarch-ls2k500/pmon/loaders/misc_bio.c
pmon-loongarch-ls2k500/pmon/loaders/loadfn.h
pmon-loongarch-ls2k500/pmon/loaders/loadfn.c
```

然后进入PMON源码目录,进入到tools/pmoncfg 下执行make pmoncfg命令,编译完成后生成文件pmoncfg,如下图所示:



```
root@ubuntu: /home/cht/work/loongarch/pmon/pmon-loongarch-ls2k500/tools/pmoncfg# ls
common.h  gram.h  hash.o  mkheaders.c  mkmakefile.c  pmoncfg  sem.c  util.o
config.h  gram.o  main.c  mkheaders.o  mkmakefile.o  pmoncfg.8  sem.h
files.c   gram.y  main.o  mkioconf.c  pack.c        scan.l    sem.o
files.o   hash.c  Makefile  mkioconf.o  pack.o        scan.o    util.c
```

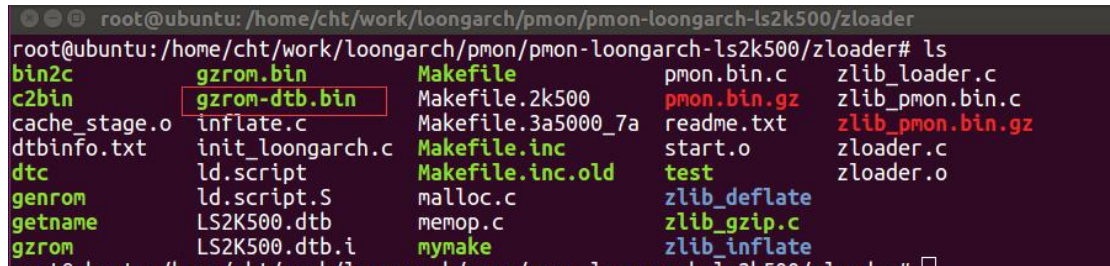
将该文件通过“cp pmoncfg /usr/bin”拷贝到/usr/bin下，然后返回PMON源码目录。创建编译脚本cmd.sh:

```
#!/bin/bash

cd zloader.2k500/
export PATH=/opt/loongarch64-linux-gnu-2022-01-26-vector/bin/:$PATH
export PATH=/opt/loongarch toolchain/bin:SPATH

make cfg all tgt=rom ARCH=loongarch CROss_COMPILE=loongarch64-linux-gnu-
DEBUG=-g
make dtb ARCH=loongarch CROss_COMPILE=loongarch64-linux-gnu-
```

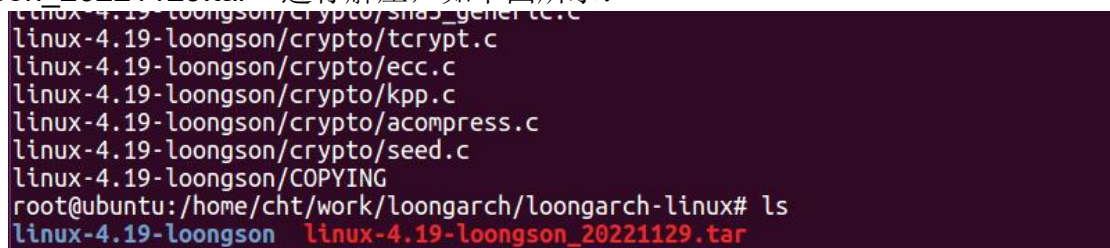
执行./cmd.sh后在目录zloader下生成文件gzrom-dtb.bin文件即为带设备树的PMON文件，如下图所示:



```
root@ubuntu: /home/cht/work/loongarch/pmon/pmon-loongarch-ls2k500/zloader# ls
bin2c          gzrom.bin      Makefile       pmon.bin.c     zlib_loader.c
c2bin          gzrom-dtb.bin  Makefile.2k500 pmon.bin.gz    zlib_pmon.bin.c
cache_stage.o  inflate.c      Makefile.3a5000_7a readme.txt     zlib_pmon.bin.gz
dtbinfo.txt    init_loongarch.c Makefile.inc    start.o        zloader.c
dtc            ld.script     Makefile.inc.old test           zloader.o
genrom         ld.script.S   malloc.c       zlib_deflate
getname        LS2K500.dtb   memop.c        zlib_gzip.c
gzrom          LS2K500.dtb.i mymake         zlib_inflate
```

6.3 编译Linux内核

拷贝 Linux 内核源码压缩包到 ubuntu 下，在终端输入“tar -xvf linux-4.19-loongson_20221129.tar”进行解压，如下图所示:



```
linux-4.19-loongson/crypto/sha3_generic.c
linux-4.19-loongson/crypto/tcrypt.c
linux-4.19-loongson/crypto/ecc.c
linux-4.19-loongson/crypto/kpp.c
linux-4.19-loongson/crypto/acompress.c
linux-4.19-loongson/crypto/seed.c
linux-4.19-loongson/COPYING
root@ubuntu: /home/cht/work/loongarch/loongarch-linux# ls
linux-4.19-loongson  linux-4.19-loongson_20221129.tar
```

进入源码目录下，依次执行下面命令:

(1) 指定交叉编译工具链:

```
export PATH=/opt/loongarch64-linux-gnu-2022-01-26-vector/bin/:$PATH
```

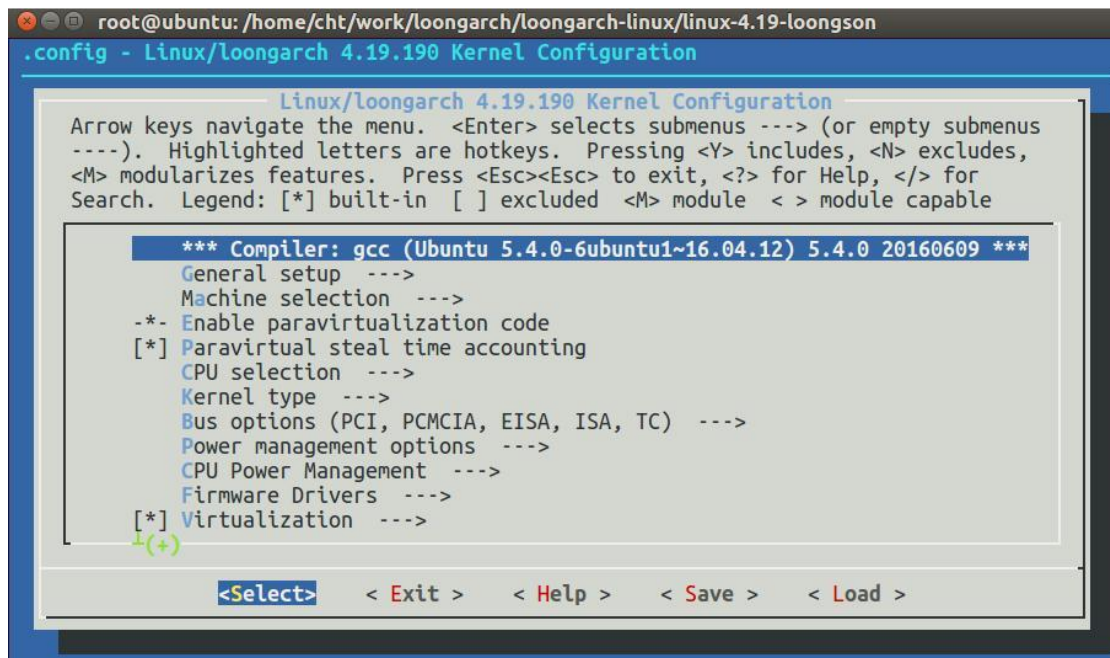
(2) 配置内核文件

```
cp config-ls2k500 .config
```

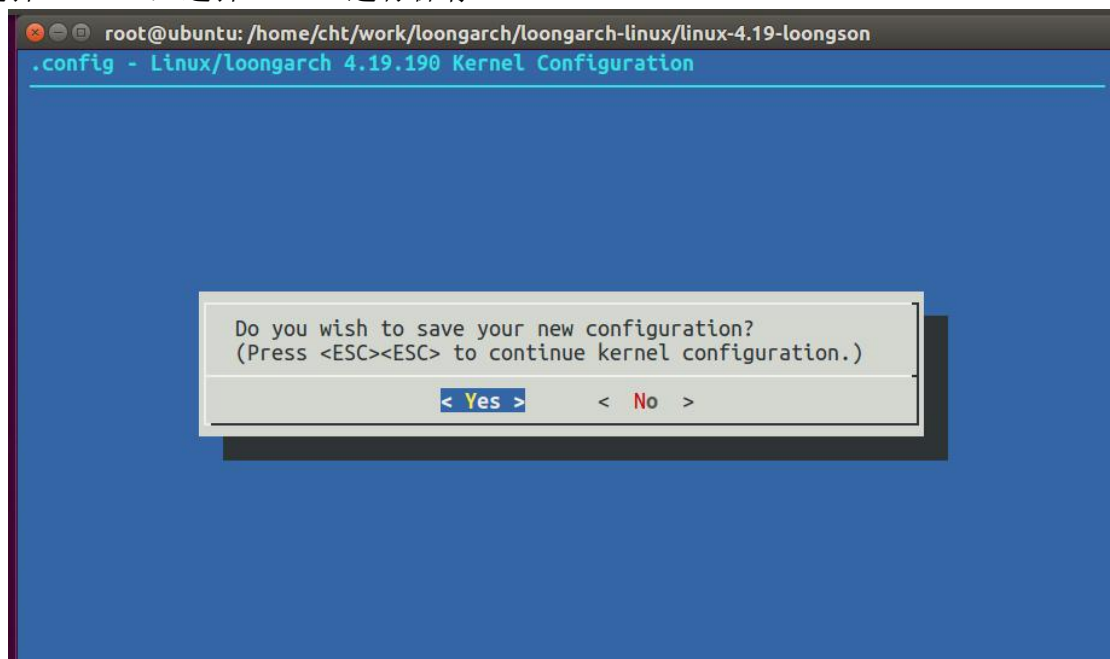
(3) 保存图形化界面配置

```
make menuconfig ARCH=loongarch
```

之后会进入到图形化界面



选择<EXIT>, 选择<YES>进行保存



(4) 编译内核

```
make vmlinuz ARCH=loongarch CROSS_COMPILE=loongarch64-linux-gnu-
```

内核编译好之后会生成一个vmlinuz的文件，这就是内核文件。

上面的编译步骤也可以修改为使用一个编译脚本完成，要确定.config文件是上面执行第二步和第三步之后保存的，否则先执行上面的步骤。创建编译脚本cmd.sh，然后将下面的写入到脚本中，在后续的编译中可以直接执行./cmd.sh即可。

```
#!/bin/bash
```

```
export PATH=/opt/loongarch64-linux-gnu-2022-01-26-vector/bin/:SPATH
```

```
make vmlinuz ARCH=loongarch CROss_cOMPILE=loongarch64-linux-gnu-
```

至此内核编译就完成了。

6.4 编译buildroot文件系统

将提供的buildroot源码解压到Ubuntu目录下，然后进入到buildroot目录下，执行以下命令。

6.4.1 配置编译器环境

```
export PATH=/opt/loongarch64-linux-gnu-2022-01-26-vector/bin/:$PATH
```

6.4.2 图形化界面配置

```
make menuconfig ARCH=loongarch64
```

使用上面的命令进行图形化界面配置，勾选自己需要的安装包进行编译，配置好后保存配置文件然后直接编译。这里类似于内核的图形化配置。

6.4.3 编译

```
make ARCH=loongarch64 CROSS_COMPILE=loongarch64-linux-gnu- -j4 "$@"
```

编译完成后会在output/images/目录下生成相关的镜像。

上面的步骤也可以直接使用脚本执行，这样在使用的时候较为方便。源码目录下存在一个cmd.sh文件，这时已经新建好的脚本文件，执行下面的命令即可：

```
./cmd.sh
```