SoC linux 系统搭建

实验目标

搭建 SoC 上的 linux 系统，正常上板运行

实验准备

百度云链接：https://pan.baidu.com/s/1ozlXbdiwz-X2yHyRFjSsSw

提取码：wkkx

源码

C5MB_GHRD 黄金参考工程

buildroot.zip

linux-socfpga.zip

gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux.tar.xz

gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux.tar.xz

软件准备

Ubuntu 14.04 虚拟机镜像

Quartus 18.1

SoC EDS

Win32DiskImager.exe

硬件

海云 AIGO_C5MB 开发板

SoC linux 系统搭建概述

1、编译硬件工程，生成 soc_system.dtb soc_system.rbf

2、编译 uboot，生成 uboot.img preloader-mkpimage.bin u-boot.scr

3、编译内核，生成 zImge

4、编译根文件系统，生成 rootfs.tar

5、制作 sd 镜像，生成 sdcard.img

6、烧写 sdcard.img 到 sd 卡

7、调试

其中，uboot、内核、文件系统是可以同时编译的，时间会少一点

一、编译 linux 内核

Ubuntu 下

解压内核源码

解压工具链

1、 切换版本到 4.9

cd linux-socfpga 进入内核目录

git tag -l 列出所有分支版本

git checkout rel_socfpga-4.9.78-ltsi_18.02.01_pr 切换版本

2、 临时指定交叉编译工具链路径

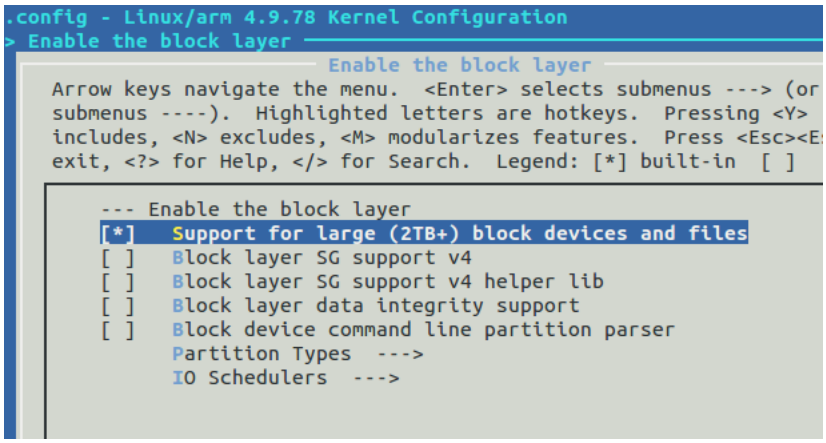
export CROSS_COMPILE=/home/z/WORK/inside_core/gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux/bin/arm-linux-gnueabihf-

3、 将内核配置为开发板出厂设置

make ARCH=arm socfpga_defconfig

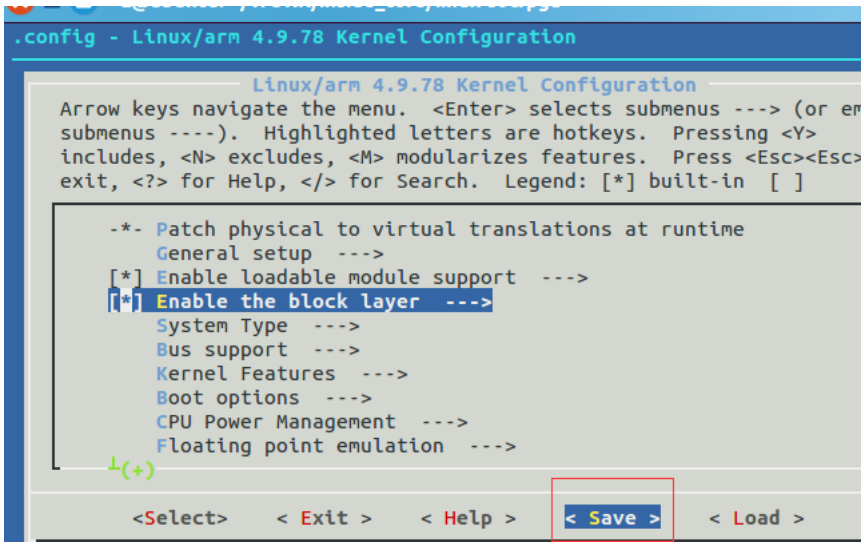
4、 修改自己的内核配置

make ARCH=arm menuconfig



一定要选择支持大存储空间设备的选项，不然生成的镜像是 read-only system

保存退出



5、 编译内核

make ARCH=arm LOCALVERSION= zImage

这个时间比较长，大概 10 分钟

```
LDS     arch/arm/boot/compressed/vmlinux.lds
AS      arch/arm/boot/compressed/head.o
GZIP    arch/arm/boot/compressed/piggy_data
AS      arch/arm/boot/compressed/piggy.o
CC      arch/arm/boot/compressed/misc.o
CC      arch/arm/boot/compressed/decompress.o
CC      arch/arm/boot/compressed/string.o
SHIPPED arch/arm/boot/compressed/hyp-stub.S
AS      arch/arm/boot/compressed/hyp-stub.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS      arch/arm/boot/compressed/lib1funcs.o
SHIPPED arch/arm/boot/compressed/ashldi3.S
AS      arch/arm/boot/compressed/ashldi3.o
SHIPPED arch/arm/boot/compressed/bswapsdi2.S
AS      arch/arm/boot/compressed/bswapsdi2.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
z@ubuntu:~/WORK/inside_core/linux-socfpga$ a
```

6、 查看生成文件

```
SHIPPED arch/arm/boot/compressed/ashldi3.S
AS      arch/arm/boot/compressed/ashldi3.o
SHIPPED arch/arm/boot/compressed/bswapsdi2.S
AS      arch/arm/boot/compressed/bswapsdi2.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
z@ubuntu:~/WORK/inside_core/linux-socfpga$ ls arch/arm/boot
bootp  compressed  dts  Image  install.sh  Makefile  zImage
z@ubuntu:~/WORK/inside_core/linux-socfpga$
```

至此内核编译完成

二、编译根文件系统

解压 buildroot

解压交叉编译工具链

1、 切换版本到 2015.08（因为这个版本支持 linaro 的 4.9-2014.09 的交叉编译器）

cd buildroot/

git checkout 2015.08

```
changes and commit them, and you can discard any commi
state without impacting any branches by performing ano

If you want to create a new branch to retain commits y
do so (now or later) by using -b with the checkout com

  git checkout -b new_branch_name

HEAD is now at 20a36cd... Update for 2015.08
```

2、 清空编译信息

make clean

```
z@ubuntu:~/WORK/inside_core/buildroot$ make clean
rm -rf /home/z/WORK/inside_core/buildroot/output/target /home/z/WORK/inside_core
/buildroot/output/images /home/z/WORK/inside_core/buildroot/output/host \
                /home/z/WORK/inside_core/buildroot/output/build /home/z/WORK/ins
ide_core/buildroot/output/staging \
                /home/z/WORK/inside_core/buildroot/output/legal-info /home/z/WOR
```

3、 配置 buildroot

cd .. 退出到上级目录

make -C buildroot ARCH=arm BR2_TOOLCHAIN_EXTERNAL_PATH=/home/z/WORK/inside_core/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux menuconfig

```
z@ubuntu:~/WORK/inside_core/buildroot$ cd ..
z@ubuntu:~/WORK/inside_core$ make -C buildroot ARCH=arm BR2_TOOLCHAIN_EXTERNAL_P
ATH=/home/z/WORK/inside_core/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux me
nuconfig
```

配置 Target Options:

在"Target Architecture"选项中，选择"ARM (little endian)"

在"Target Architecture variant"选项中，选中 "cortex-A9"

在"Target ABI"选项中， 选中"EABIhf"

Enable “NEON SIMD extension support”

在“Floating point strategy”选项中， 选中"NEON"

“Target Binary Format” and “ARM Instruction set” 选项保持默认。

配置 Toolchain:

在“Toolchain type”选项中， 选中 “External toolchain”

确保 “Toolchain” 选项中，选中 “Linaro ARM 2014.09”.

在“Toolchain origin”选项中， 选中 “Pre-installed toolchain”

忽略"toolchain path"

Enable “copy gdb server to the Target”

其他选项保持默认

配置 System configuration:

配置 hostname

配置 root password

配置 Kernel:

去掉“Linux Kernel” 选项的选中状态

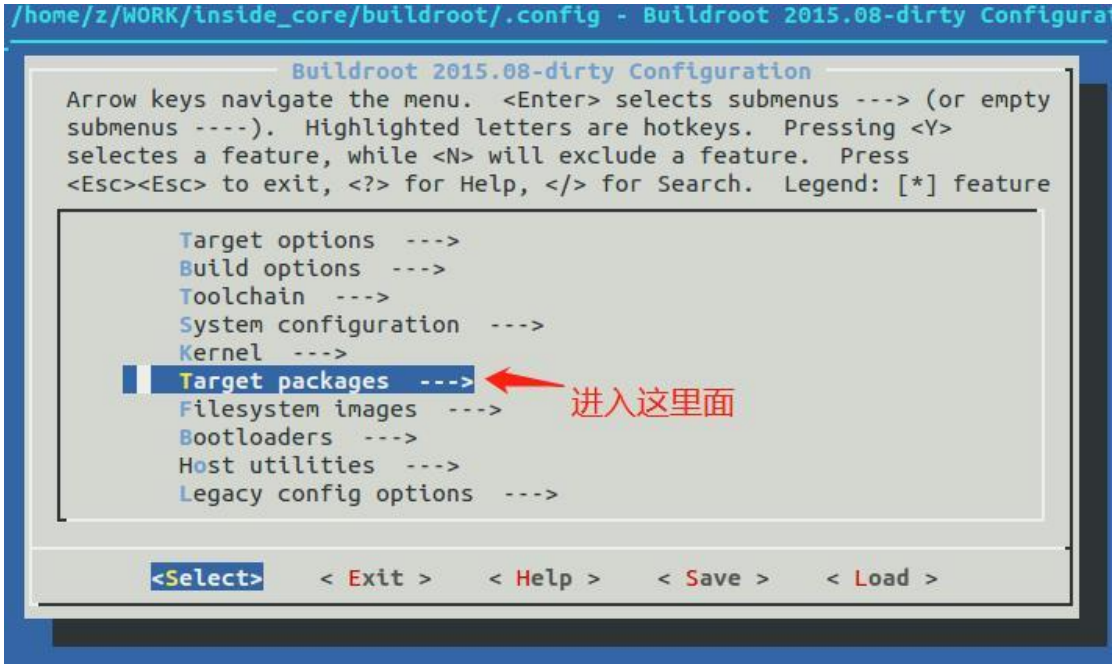
配置 Target packages

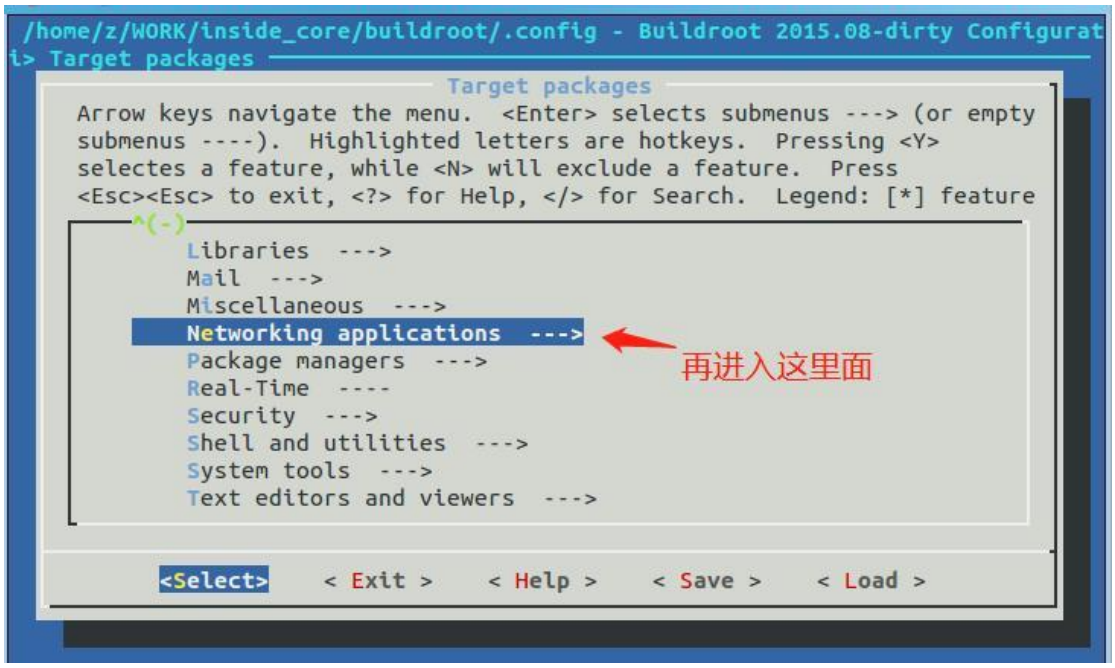
在 “Debugging, profiling and benchmark”选项中，拖动滚动条到底部并选中 “valgrind”

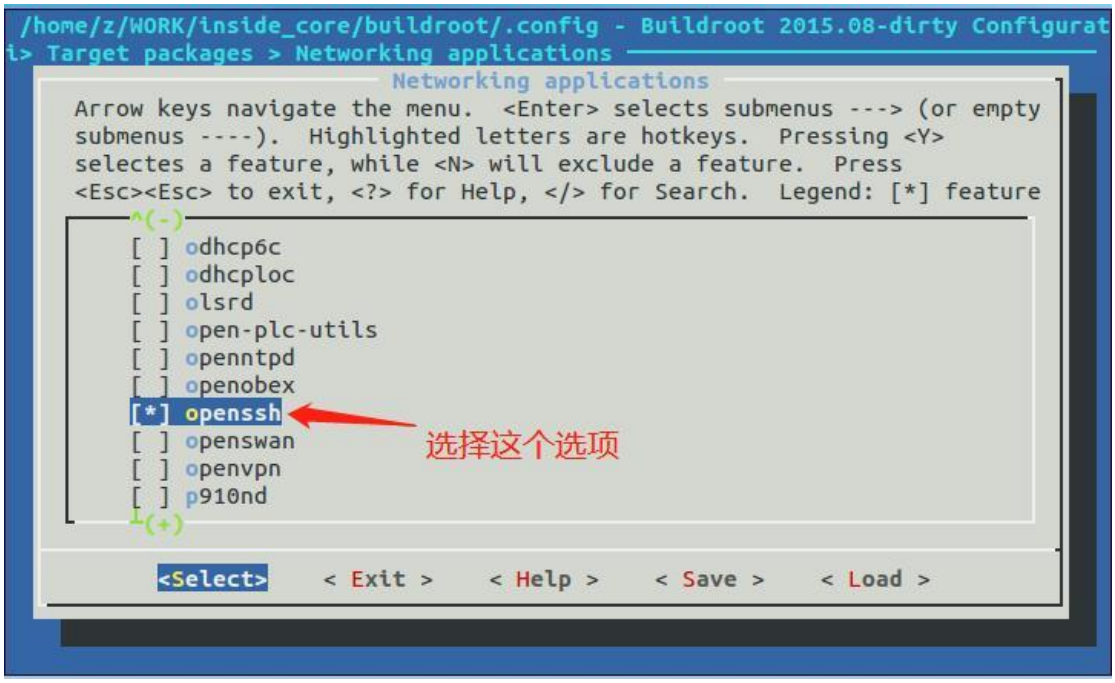
其它

按照自己需求选定应用软件，比如 openssh

4、 配置 ssh


/home/z/WORK/inside_core/buildroot/.config - Buildroot 2015.08-dirty Configurat
Buildroot 2015.08-dirty Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selectes a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature
Target options --->
Build options --->
Toolchain --->
System configuration --->
Kernel --->
Target packages --->
Filesystem images --->
Bootloaders --->
Host utilities --->
Legacy config options --->
进入这里面
<Select> < Exit > < Help > < Save > < Load >


/home/z/WORK/inside_core/buildroot/.config - Buildroot 2015.08-dirty Configurat
i> Target packages
Target packages
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selectes a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature
^(-)
Libraries --->
Mail --->
Miscellaneous --->
Networking applications --->
Package managers --->
Real-Time ----
Security --->
Shell and utilities --->
System tools --->
Text editors and viewers --->
再进入这里面
<Select> < Exit > < Help > < Save > < Load >

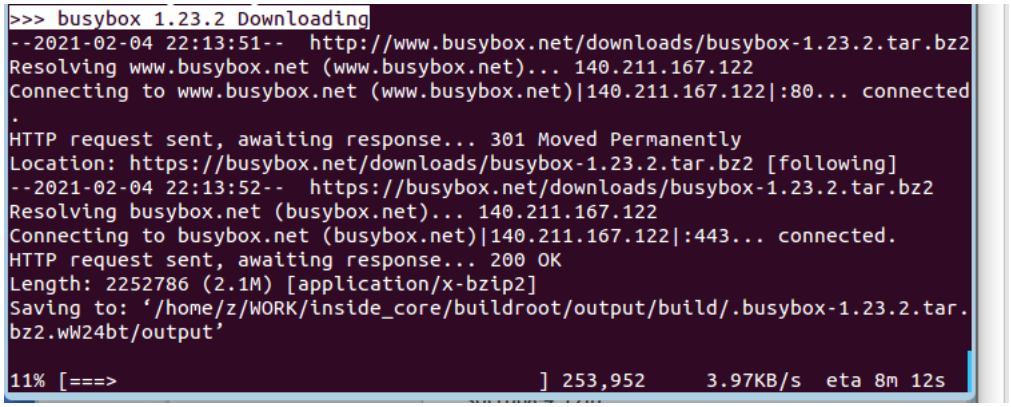


保存退出。

5、 配置 busybox

make -C buildroot busybox-menuconfig

这一步需要下载文件，所以要确保网络连通能用，时间需要几分钟根据网速来的

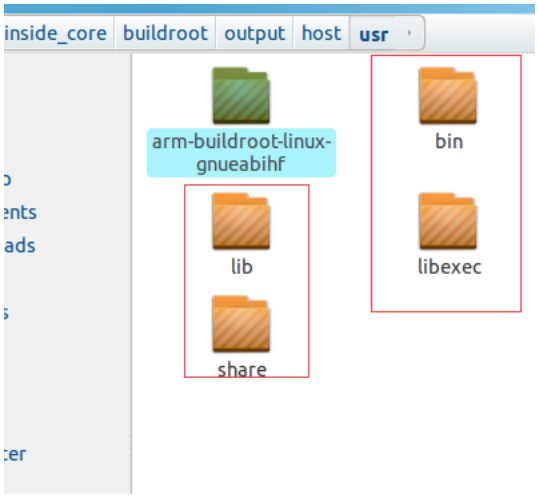


Busybox Settings->Build Options->Build BusyBox as a static binary (no shared libs)

是否使用静态编译，如果不是静态编译，则 busybox 运行时还需要复制几个动态库文件，否则不能运行，但是实际上我尝试了用静态编译，会出现无法 login 的情况。所以这里不用静态编译

这里面会报错缺少文件

复制交叉编译工具链里面的下面几个文件夹到 buildroot/output/host/usr 目录下



再次配置

make -C buildroot busybox-menuconfig

退出

6、 编译根文件系统

make -C buildroot BR2_TOOLCHAIN_EXTERNAL_PATH=/home/z/WORK/inside_core/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux all

这一步需要下载文件，所以要保持网络畅通，时间的话需要几十分钟，网速快时间会稍微短一些，可以去喝两杯咖啡

```
RK/inside_core/buildroot/output/build/_fakeroot.fs
chmod a+x /home/z/WORK/inside_core/buildroot/output/build/_fakeroot.fs
PATH="/home/z/WORK/inside_core/buildroot/output/host/bin:/home/z/WORK/inside_cor
e/buildroot/output/host/sbin:/home/z/WORK/inside_core/buildroot/output/host/usr/
bin:/home/z/WORK/inside_core/buildroot/output/host/usr/sbin:/usr/local/sbin:/usr
/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/home/z/pro
gramm/bin" /home/z/WORK/inside_core/buildroot/output/host/usr/bin/fakeroot -- /h
ome/z/WORK/inside_core/buildroot/output/build/_fakeroot.fs
rootdir=/home/z/WORK/inside_core/buildroot/output/target
table='/home/z/WORK/inside_core/buildroot/output/build/_device_table.txt'
/usr/bin/install -m 0644 support/misc/target-dir-warning.txt /home/z/WORK/inside
_core/buildroot/output/target/THIS_IS_NOT_YOUR_ROOT_FILESYSTEM
make: Leaving directory `/home/z/WORK/inside_core/buildroot'
z@ubuntu:~/WORK/inside_core$
```

另外编译有错，一般是网络不好，重新开始就行

7、 查看生成文件

```
z@ubuntu:~/WORK/inside_core$ ls buildroot/output/images/
rootfs.tar
z@ubuntu:~/WORK/inside_core$
```

三、编译 uboot

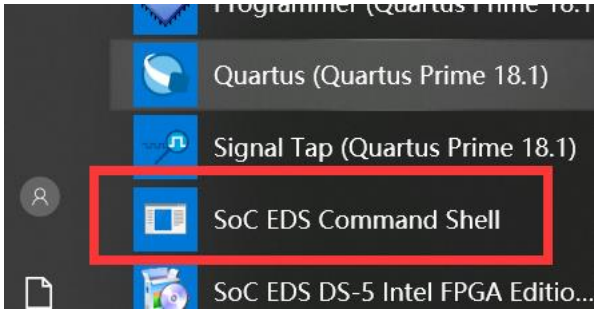
需要黄金参考工程 C5MB_GHRD

这个可以在 windows 下也可以在 ubuntu 下

Windows 下

1、更新硬件信息

打开 eds



切换到工程目录

```
AWcloud@DESKTOP-I6BGBA6
$ cd Desktop/soc_project/C5MB_GHRD

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD
$
```

生成设备树

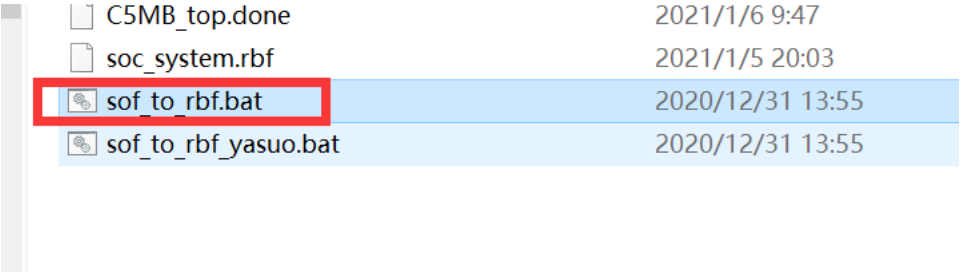
```
AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD
$ make dtb
```

或者

输入 make dts ， 生成 soc_system.dts 设备树文件

输入 make dtb ， 生成 soc_system.dtb 二进制格式的设备树文件

生成 rbf 文件

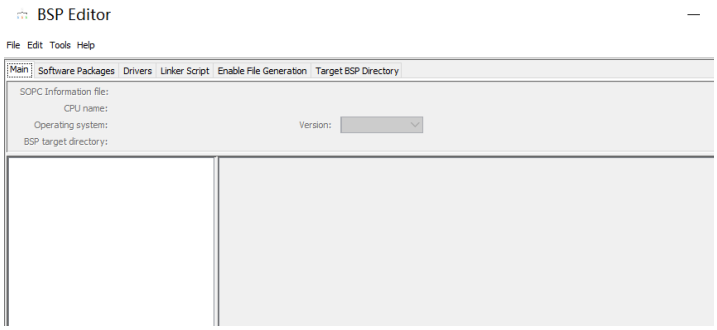


C5MB_top.done	2021/1/6 9:47
soc_system.rbf	2021/1/5 20:03
sof_to_rbf.bat	2020/12/31 13:55
sof_to_rbf_yasuo.bat	2020/12/31 13:55

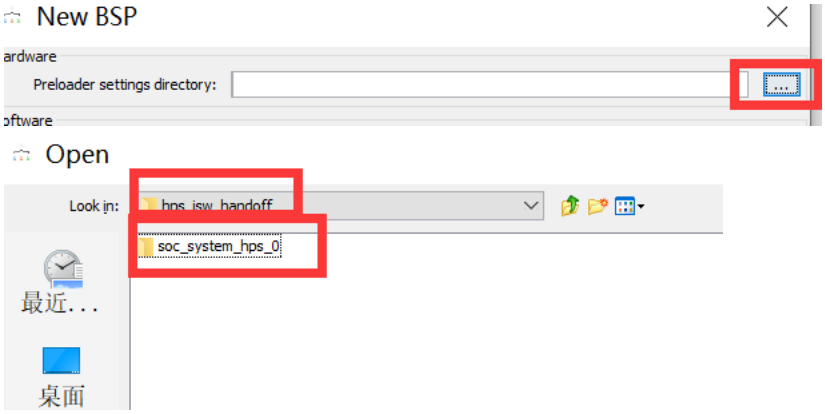
2、生成 u-boot.img 文件

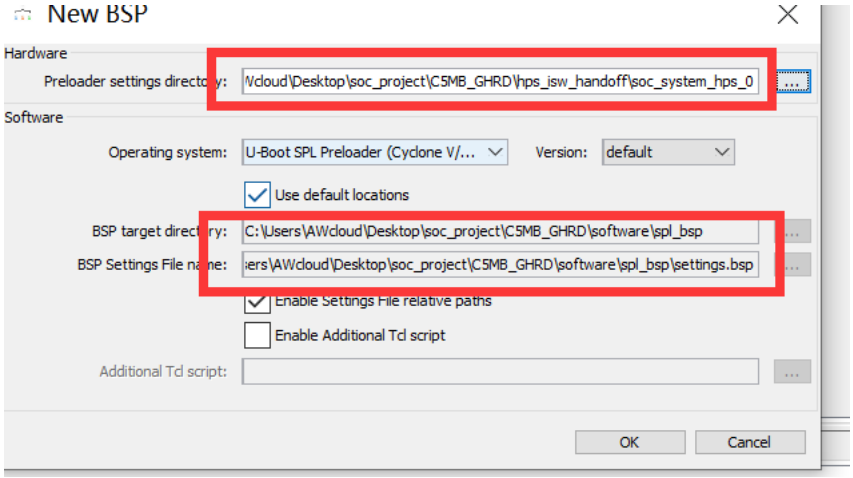
生成 bsp

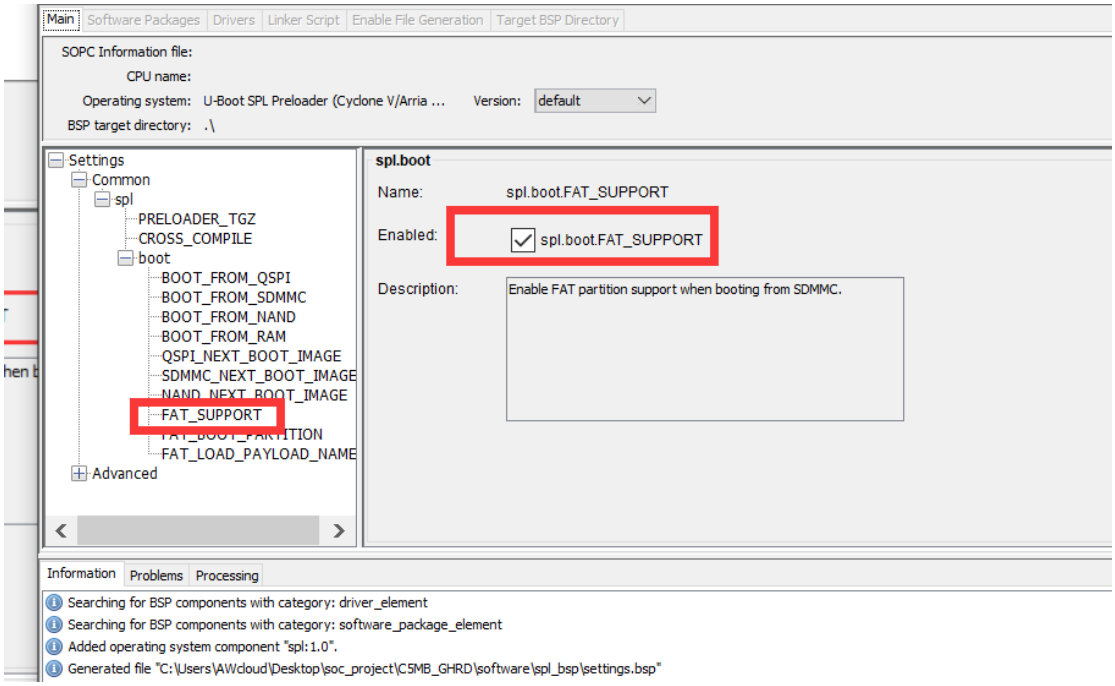
```
AWcloud@DESKTOP-I6BGBA
$ cd software/

AWcloud@DESKTOP-I6BGBA
$ bsp-editor.exe
```



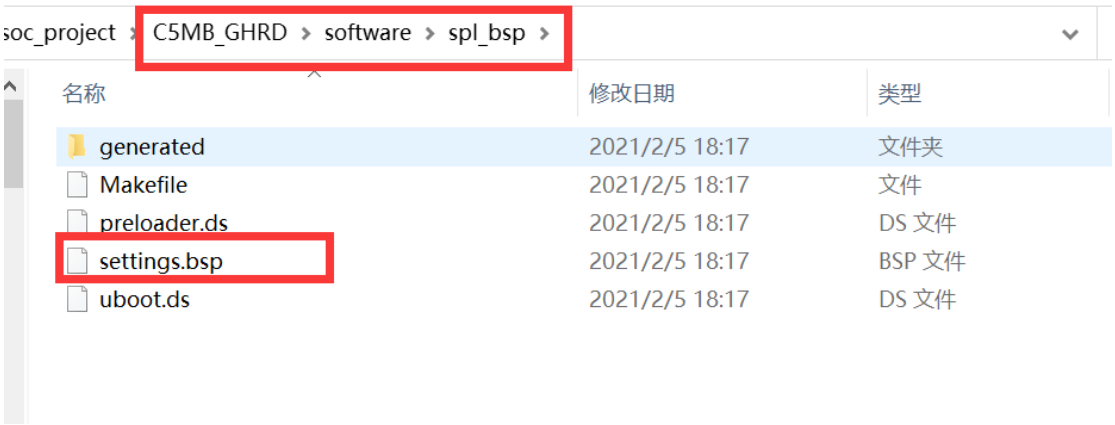
点击 file -> new HPS BSP







生成 bsp



进入 sp1_bsp 文件夹

编译

```
AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software
$ cd spl_bsp/

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp
$ ls
generated  Makefile  preloader.ds  settings.bsp  uboot.ds

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp
$ make uboot
tar zxf /cygdrive/d/intelFPGA/18.1/embedded/host_tools/altera/preloader/uboot-socfpga
tar: Error opening archive: Failed to open '/cygdrive/d/intelFPGA/18.1/embedded/host_
a.tar.gz'
make: *** [uboot-socfpga/.untar] Error 1
```

替换 quartus 下面的文件

设置环境变量

export PATH=/bin:$PATH

```
AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp
$ export PATH=/bin:$PATH
```

再次编译 生成，这个时间大概 10 分钟

make uboot

```
                -Ld:/intelfpga/18.1/embedded/host_tools/mentor/gnu/arm/baremetal/bin/../lib/gcc/arm-altera-eabi/6.2.0 -
gcc
arm-altera-eabi-objcopy -O srec hello_world hello_world.srec 2>/dev/null
arm-altera-eabi-objcopy -O binary hello_world hello_world.bin 2>/dev/null
make[2]: Leaving directory '/cygdrive/c/Users/AWcloud/Desktop/soc_project/C5MB_GHRD/software/spl_bsp/uboot-socfpga/exam
les/standalone'
/bin/make -C examples/api all
make[2]: Entering directory '/cygdrive/c/Users/AWcloud/Desktop/soc_project/C5MB_GHRD/software/spl_bsp/uboot-socfpga/exa
ples/api'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/cygdrive/c/Users/AWcloud/Desktop/soc_project/C5MB_GHRD/software/spl_bsp/uboot-socfpga/exam
les/api'
make[1]: Leaving directory '/cygdrive/c/Users/AWcloud/Desktop/soc_project/C5MB_GHRD/software/spl_bsp/uboot-socfpga'

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp
$
```

soc_project › C5MB_GHRD › software › spl_bsp › uboot-socfpga ›

名称	修改日期	类型
rules.mk	2021/2/5 18:26	Makefile
snapshot.commit	2018/9/13 23:22	COMMIT 文件
System.map	2021/2/5 18:35	Linker Address M
u-boot	2021/2/5 18:35	文件
u-boot.bin	2021/2/5 18:35	BIN 文件

soc_project › C5MB_GHRD › software › spl_bsp › uboot-socfpga › spl ›

名称	修改日期	类型	大小
spl	2021/2/5 18:35	文件夹	
.depend	2021/2/5 18:35	DEPEND 文件	0 KB
.gitignore	2018/9/13 23:22	Git Ignore 源文件	1 KB
Makefile	2018/9/13 23:22	文件	6 KB
u-boot.lst	2021/2/5 18:37	MASM Listing	0 KB
u-boot-spl	2021/2/5 18:37	文件	654 KB
u-boot-spl.bin	2021/2/5 18:37	BIN 文件	42 KB

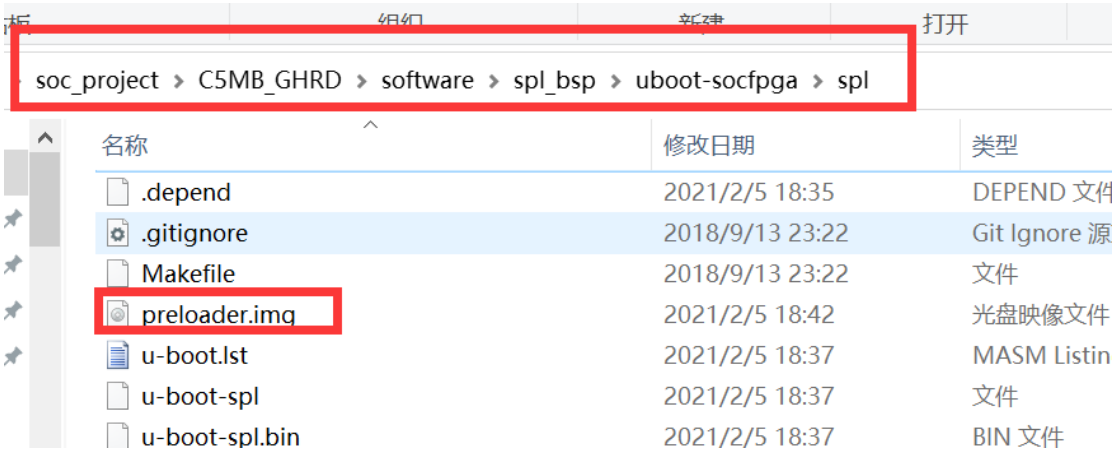
3、生成 preloader-mkpimage.bin 文件

cd uboot-socfpga/spl/

mkpimage -hv 0 -o preloader.img u-boot-spl.bin

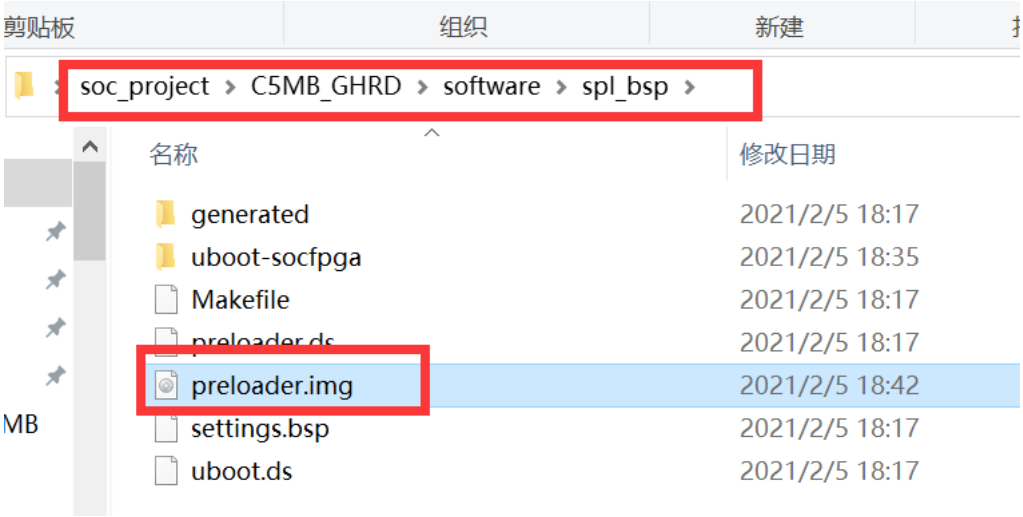
```
$

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp
$ cd uboot-socfpga/spl/

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp/uboot-soc
$ mkpimage -hv 0 -o preloader.img u-boot-spl.bin

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp/uboot-soc
```

查看生成文件



5、生成对应 bin 文件

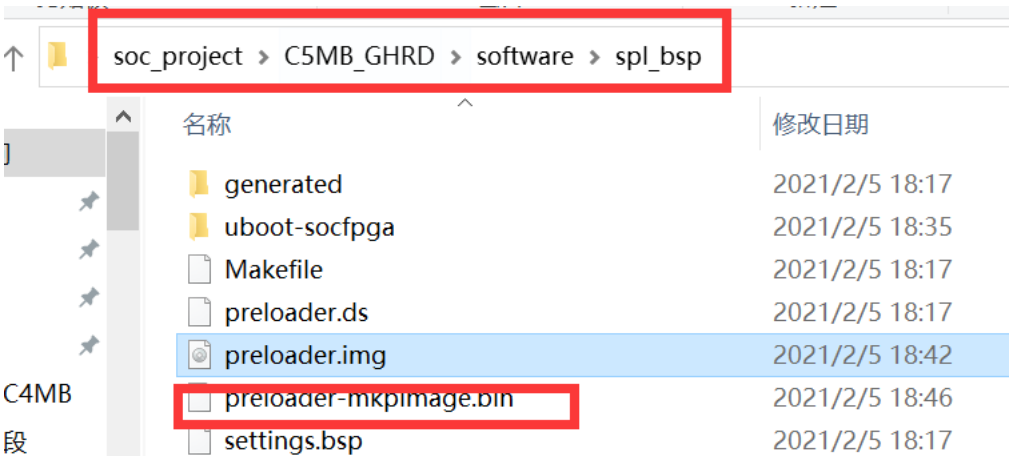
复制 preloader.img 文件到 softaware->spl_bsp 目录下



输入 make，将 preloader.img 文件编译成二进制格式的 preloader-mkpimage.bin 文件

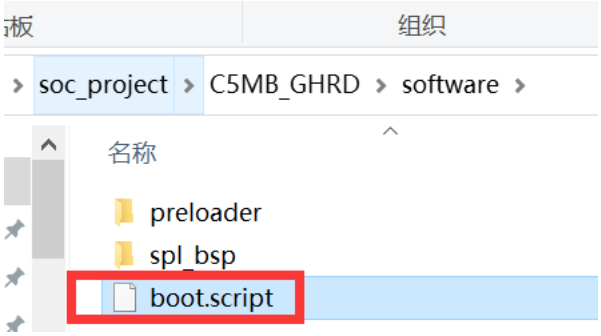
```
AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp/uboot-soc
$ cd ../..

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp
$ make
mkpimage --header-version 0 -o preloader-mkpimage.bin uboot-socfpga/spl/u-boot-spl
uboot-socfpga/spl/u-boot-spl.bin uboot-socfpga/spl/u-boot-spl.bin
```

查看生成文件



6、生成 u-boot.scr 文件

在 software 目录下创建 boot.script 文件



写入

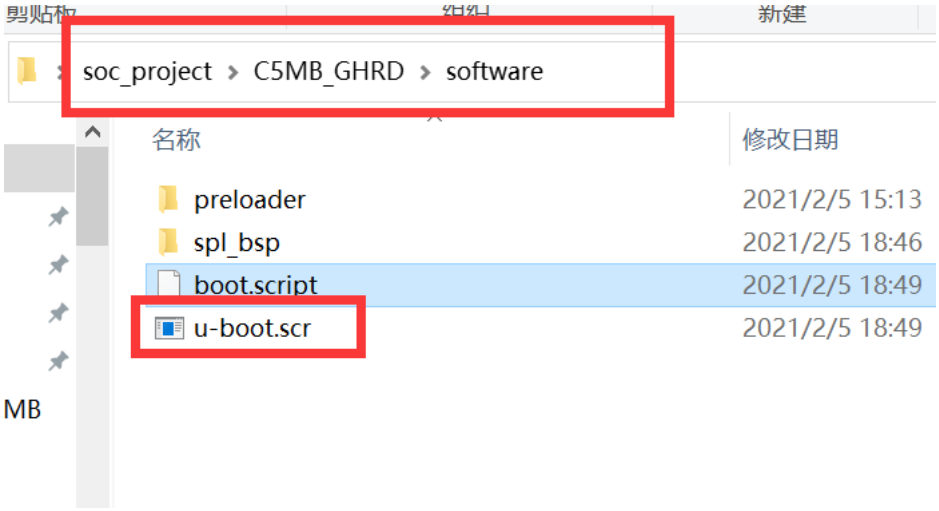
```
echo -- Programming FPGA --
fatload mmc 0:1 $fpgadata soc_system.rbf;
fpga load 0 $fpgadata $filesize;
run bridge_enable_handoff;

echo -- Setting Env Variables --
setenv fdtimage soc_system.dtb;
setenv mmcroot /dev/mmcblk0p2;
setenv mmcload 'mmc rescan;${mmcloadcmd} mmc 0:${mmcloadpart} ${loadaddr} ${bootimage};${mmcloadcmd} mmc 0:${mmcloadpart} ${fdtaddr} ${fdtimage};';
setenv mmcboot 'setenv bootargs console=ttyS0,115200 root=${mmcroot} rw rootwait; bootz ${loadaddr} - ${fdtaddr}';

run mmcload;
run mmcboot;
```

生成文件

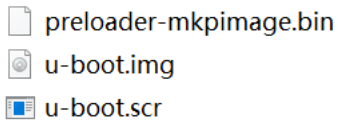
```
mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n "Boot Script Name" -d boot.script u-boot.scr
```

```
AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software/spl_bsp
$ cd ..

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software
$ mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n "Boot Script Name"
Image Name:   Boot Script Name
Created:      Fri Feb 05 18:49:36 2021
Image Type:   ARM Linux Script (uncompressed)
Data Size:    536 Bytes = 0.52 kB = 0.00 MB
Load Address: 00000000
Entry Point:  00000000
Contents:
   Image 0: 528 Bytes = 0.52 kB = 0.00 MB

AWcloud@DESKTOP-I6BGBA6 ~/Desktop/soc_project/C5MB_GHRD/software
```

查看生成文件



整合 uboot 文件
C5MB_GHRD\software\spl_bsp\uboot-socfpga\u-boot.img
C5MB_GHRD\software\u-boot.scr
C5MB_GHRD\software\spl_bsp\preloader-mkpimage.bin

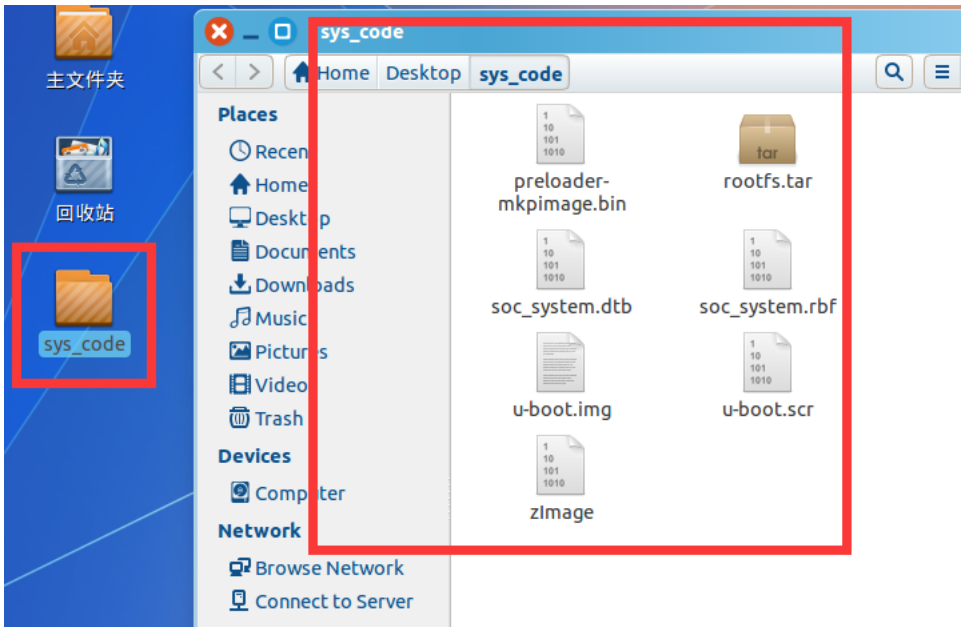


至此 uboot 所有文件都已生成

Ubuntu 下
解压 uboot 源码
预留编写

四、制作镜像文件
镜像生成分区，将内核、文件系统以及 uboot 和硬件配置放到镜像里面

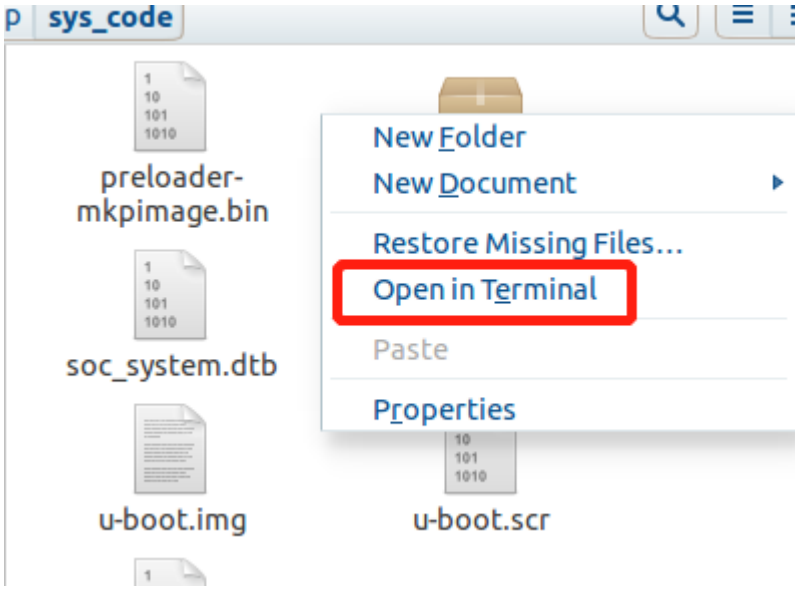
文件准备
硬件文件
黄金工程/ soc_system.dtb
黄金工程/output_files/soc_system.rbf
Uboot:
黄金工程/software/u-boot.scr
黄金工程/software/spl_bsp/preloader-mkpimage.bin
黄金工程/software/u-boot-socfpga/u-boot.img
内核:
linux-socfpga/arch/arm/boot/zImage
文件系统
buildroot/output/images/rootfs.tar
整合到一起，拖到 ubuntu 里面去



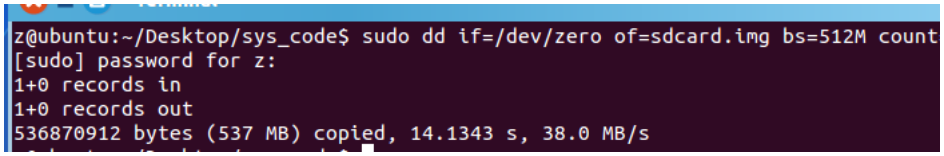
一个完整的 SOC 板上系统分为 3 个分区： FAT32、 EXT3、 RAW(A2)，每个分区保存的文件如下：

SoC FPGA Linux 系统镜像分区	每个分区的文件
RAW(A2)	preloader-mkpimage.bin、u-boot.img
EXT4	rootfs
FAT	u-boot.scr、soc_system.dtb、soc_system.rbf、zImage、u-boot.img

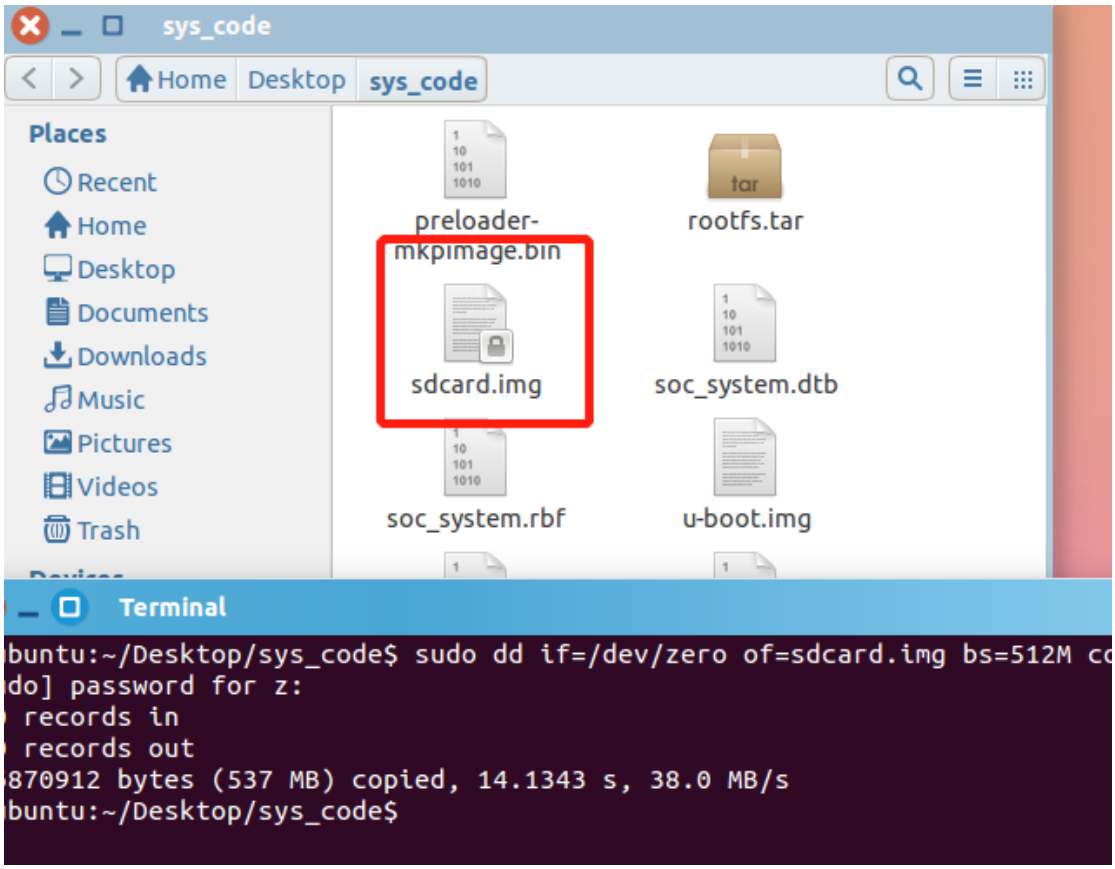
1、制作 512M 的虚空镜像
在文件夹里面打开终端



sudo dd if=/dev/zero of=sdcard.img bs=512M count=1



产生一个 img 文件



获取 loop device 设备名

sudo losetup --show -f sdcard.img

```
z@ubuntu:~/Desktop/sys_code$ sudo dd if=/dev/zero of=sdcard.img bs=
[sudo] password for z:
1+0 records in
1+0 records out
536870912 bytes (537 MB) copied, 14.1343 s, 38.0 MB/s
z@ubuntu:~/Desktop/sys_code$ sudo losetup --show -f sdcard.img
/dev/loop0
```

对 sdcard.img 进行分区：

sudo fdisk /dev/loop0

输入指令：p，查看 sdcard.img 的分区情况：目前 sdcard.img 没有任何分区。

在 sdcard.img 上创建分区 3，用来存储 preloader image，相应的配置信息如下：

n p 3 回车 +1M

```
z@ubuntu:~/Desktop/sys_code$ sudo fdisk /dev/loop0
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF diskla
el
Building a new DOS disklabel with disk identifier 0x83abf0bf.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 3
First sector (2048-1048575, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-1048575, default 1048575): +1M
```

转换分区类型为“a2”

t a2

```
Command (m for help): t
Selected partition 3
Hex code (type L to list codes): a2
Changed system type of partition 3 to a2 (Unknown)

Command (m for help): ^[^A
```

创建分区 2，用来存放 Linux root 文件系统，相应的配置信息如下，不需要进行分区类型转换：

n p 2 回车 +254M

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 2
First sector (4096-1048575, default 4096):
Using default value 4096
Last sector, +sectors or +size{K,M,G} (4096-1048575, default 1048575): +254M

Command (m for help):
```

创建分区 1，用来存放启动文件：相应的配置信息如下：

n p 1 回车 回车

```
Command (m for help): n
Partition type:
   p   primary (2 primary, 0 extended, 2 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (524288-1048575, default 524288):
Using default value 524288
Last sector, +sectors or +size{K,M,G} (524288-1048575, default 1048575):
Using default value 1048575

Command (m for help):
```

转换分区 1 分区类型为 FAT:

t 1 b

```
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): b
Changed system type of partition 1 to b (W95 FAT32)
```

输入指令：p，查看 sdcard.img 的分区情况：

```
Command (m for help): p

Disk /dev/loop0: 536 MB, 536870912 bytes
255 heads, 63 sectors/track, 65 cylinders, total 1048576 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x83abf0bf

      Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1           524288     1048575      262144    b  W95 FAT32
/dev/loop0p2             4096      524287      260096   83  Linux
/dev/loop0p3             2048        4095        1024   a2  Unknown

Partition table entries are not in disk order
```

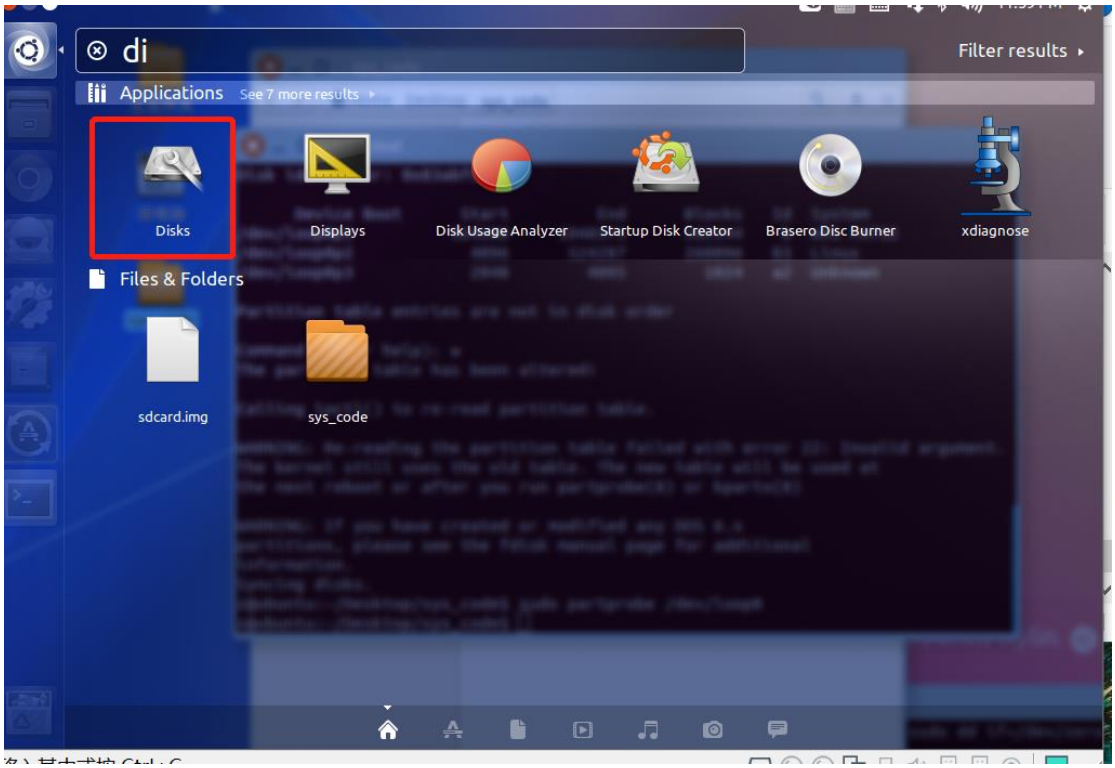
输入指令 w，退出分区操作。请无视警告信息。

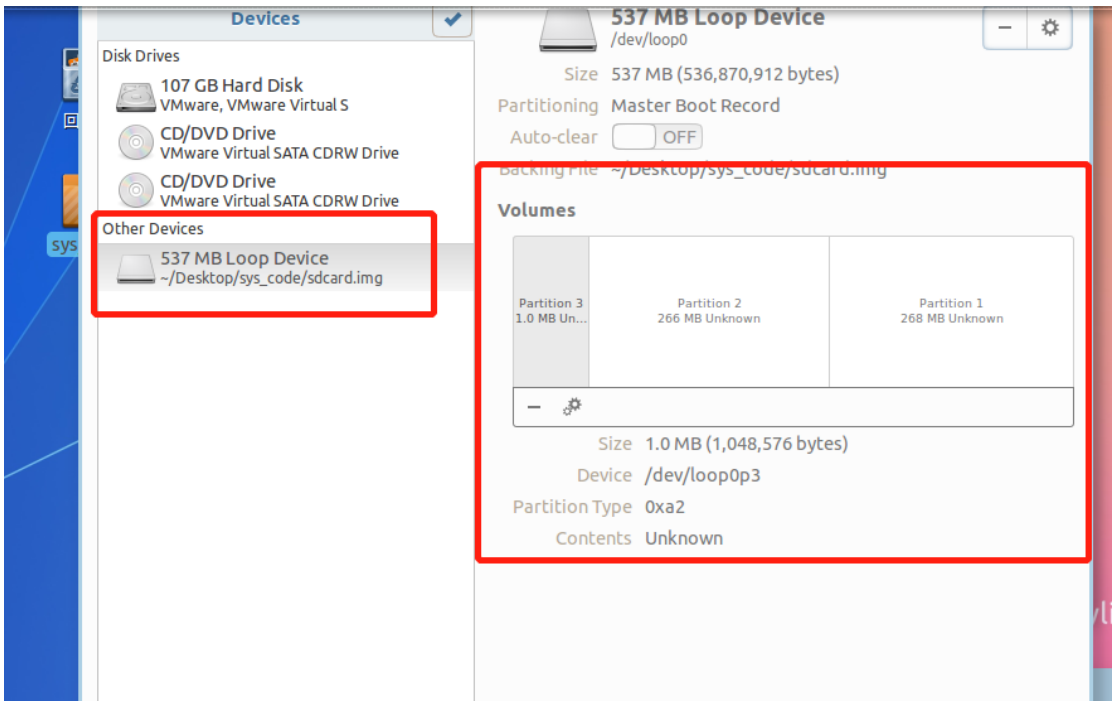
```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 22: Invalid ar
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
z@ubuntu:~/Desktop/sys_code$
```

分区还未生效，输入 partprobe 指令，让新分区生效。

sudo partprobe /dev/loop0

可以看看镜像的分区如下





格式化分区 1 为 FAT 文件系统

sudo mkfs -t vfat /dev/loop0p1

格式化分区 2 为 ext4 文件系统

sudo mkfs.ext4 /dev/loop0p2

```
z@ubuntu:~/Desktop/sys_code$ sudo mkfs -t vfat /dev/loop0p1
mkfs.fat 3.0.26 (2014-03-07)
unable to get drive geometry, using default 255/63
z@ubuntu:~/Desktop/sys_code$ sudo mkfs.ext4 /dev/loop0p2
mke2fs 1.42.9 (4-Feb-2014)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
65024 inodes, 260096 blocks
13004 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67371008
32 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
```

2、写入文件到镜像

1）写入分区 3 文件

将 preloader-mkpimage.bin 和 u-boot.img 写入分区 3

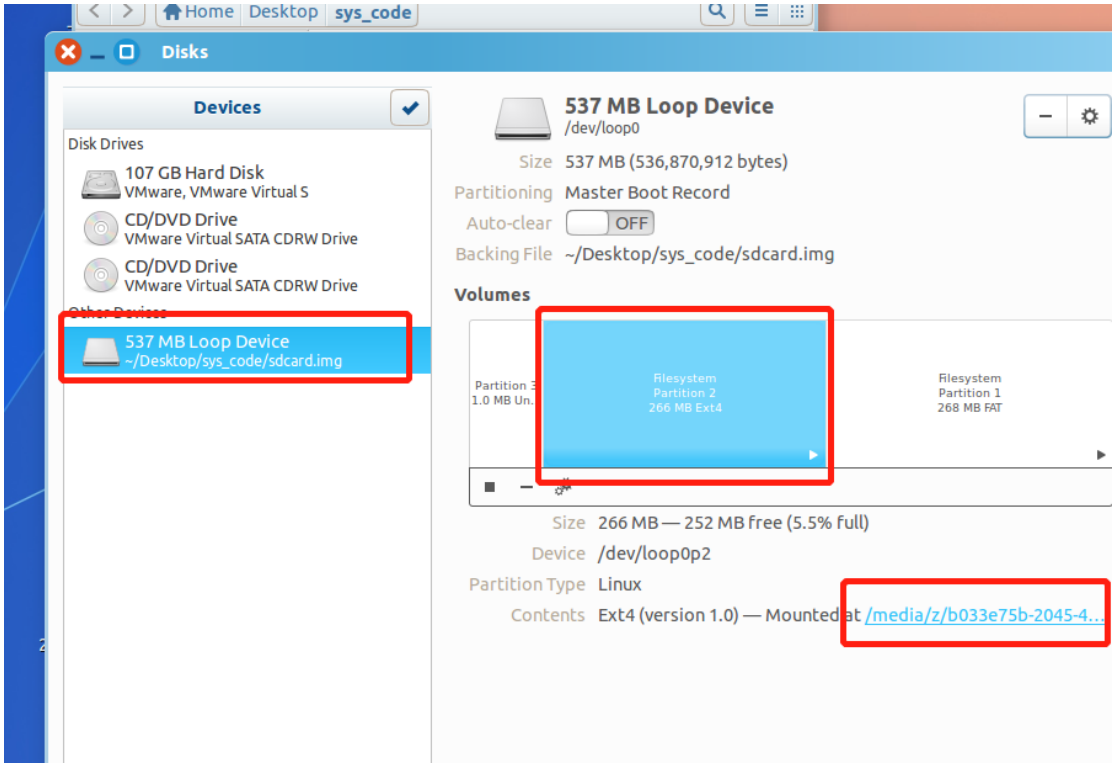
sudo dd if=preloader-mkpimage.bin of=/dev/loop0p3 bs=64k seek=0

sudo dd if=u-boot.img of=/dev/loop0p3 bs=64k seek=4

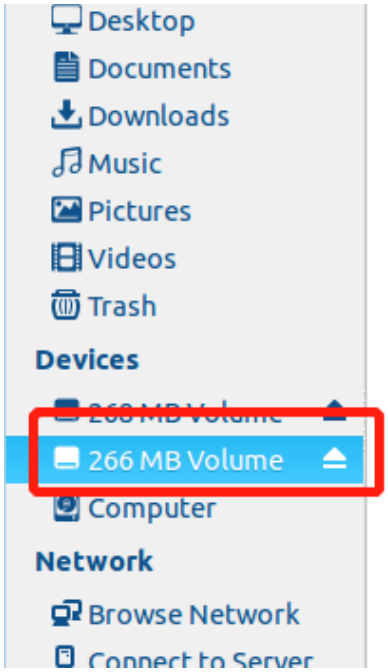
```
z@ubuntu:~/Desktop/sys_code$ sudo dd if=preloader-mkpimage.bin of=/dev/loop0p3 b
s=64k seek=0
4+0 records in
4+0 records out
262144 bytes (262 kB) copied, 0.00365561 s, 71.7 MB/s
z@ubuntu:~/Desktop/sys_code$ sudo dd if=u-boot.img of=/dev/loop0p3 bs=64k seek=4
3+1 records in
3+1 records out
237748 bytes (238 kB) copied, 0.00317228 s, 74.9 MB/s
z@ubuntu:~/Desktop/sys_code$
```

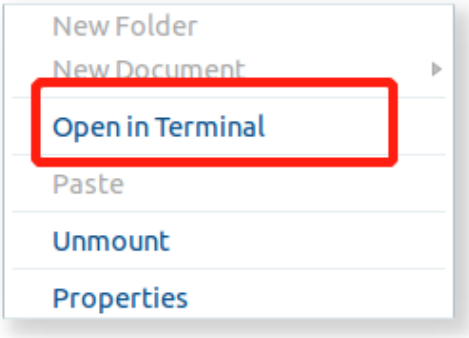
注意：/dev/loop0p3 是我的 Loop Device 的分区 3，这个“loop0p3”在每个机器上不一样，请在 disk 工具中查看清楚后再操作。

2）写入分区 2 文件，文件系统



在分区 2 打开终端



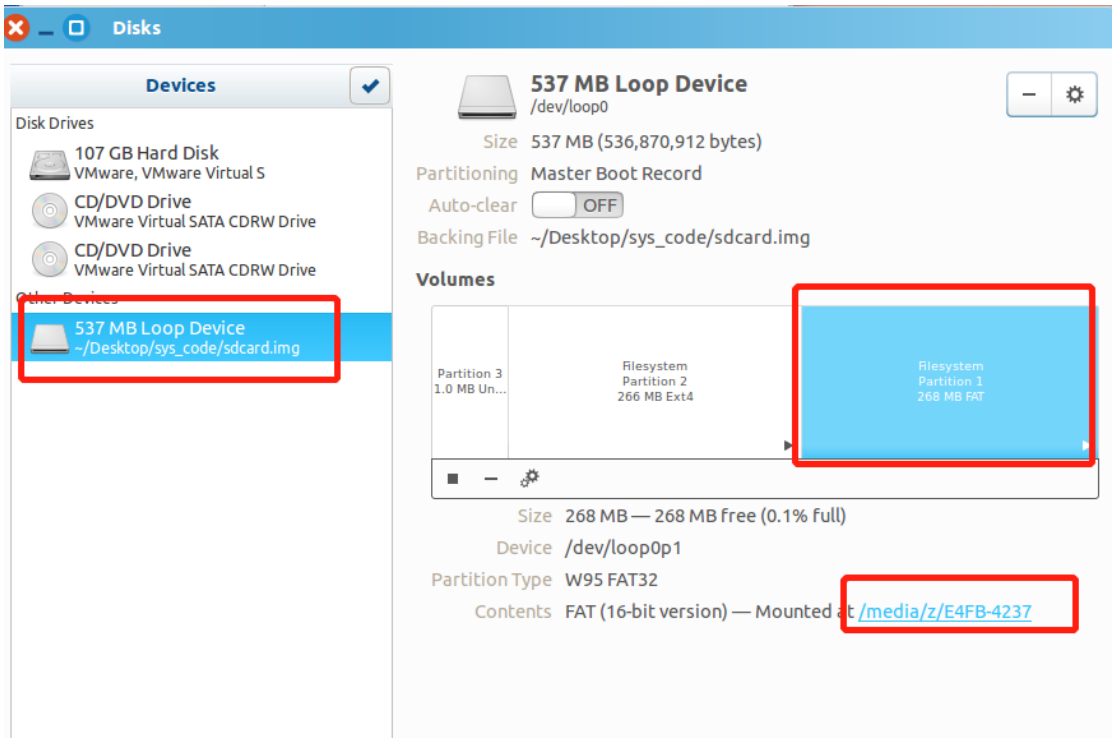


解压文件系统到分区 2

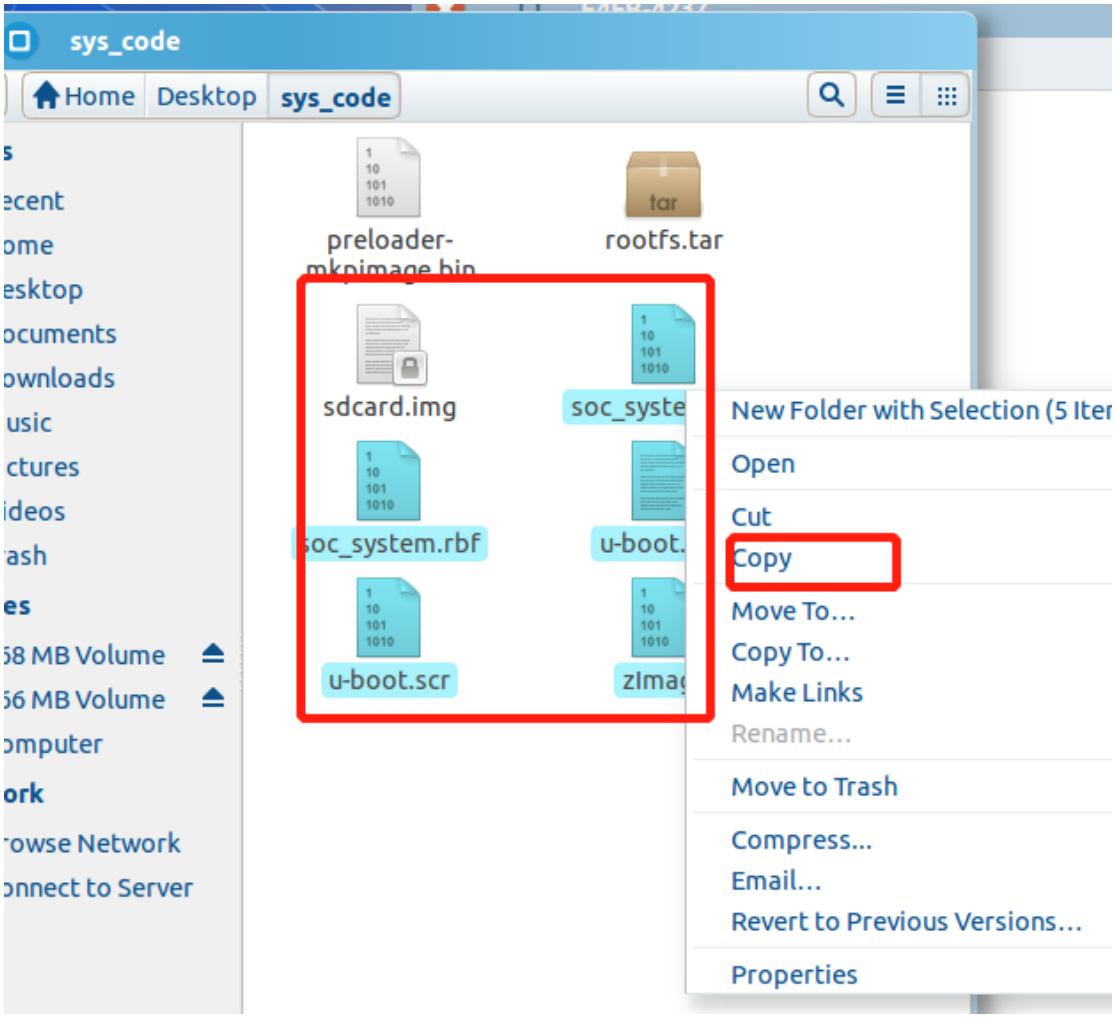
sudo tar -xf ~/Desktop/sys_code/rootfs.tar -C ./

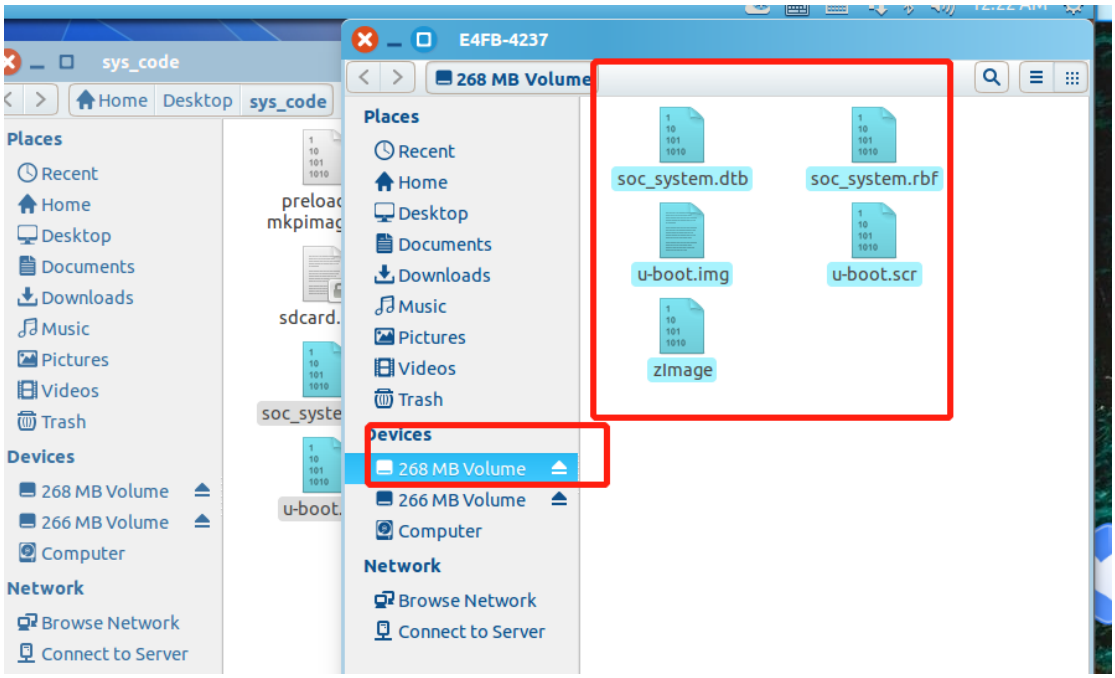
```
z@ubuntu:/media/z/b033e75b-2045-4830-be15-3ab756ca26ec$ sudo tar -xf ~/Desktop/s
ys_code/rootfs.tar -C ./
[sudo] password for z:
z@ubuntu:/media/z/b033e75b-2045-4830-be15-3ab756ca26ec$
```

3）写入分区 1 文件，


Disks
Devices
Disk Drives
107 GB Hard Disk
VMware, VMware Virtual S
CD/DVD Drive
VMware Virtual SATA CDRW Drive
CD/DVD Drive
VMware Virtual SATA CDRW Drive
537 MB Loop Device
~/Desktop/sys_code/sdcard.img
537 MB Loop Device
/dev/loop0
Size 537 MB (536,870,912 bytes)
Partitioning Master Boot Record
Auto-clear OFF
Backing File ~/Desktop/sys_code/sdcard.img
Volumes
Partition 3
1.0 MB Un...
Filesystem
Partition 2
266 MB Ext4
Filesystem
Partition 1
268 MB FAT
Size 268 MB — 268 MB free (0.1% full)
Device /dev/loop0p1
Partition Type W95 FAT32
Contents FAT (16-bit version) — Mounted at /media/z/E4FB-4237

复制文件到，镜像

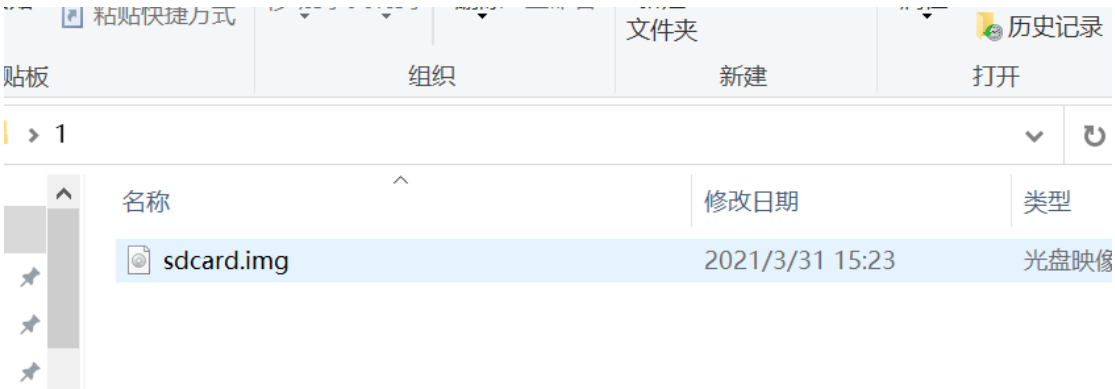

sys_code
Home
Desktop
sys_code
preloader-
rootfs.tar
sdcard.img
soc_system.rbf
u-boot.scr
New Folder with Selection (5 Iter
Open
Cut
Copy
Move To...
Copy To...
Make Links
Rename...
Move to Trash
Compress...
Email...
Revert to Previous Versions...
Properties



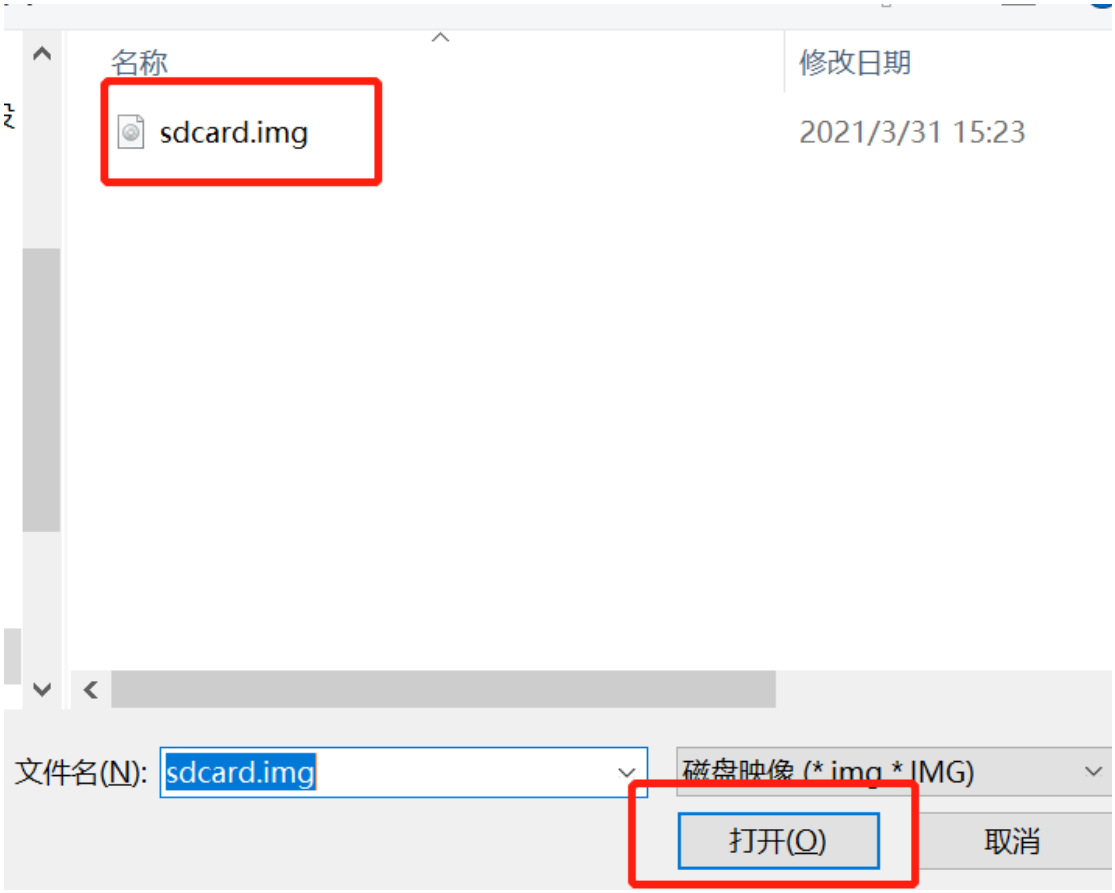
至此，sd 卡镜像制作完成

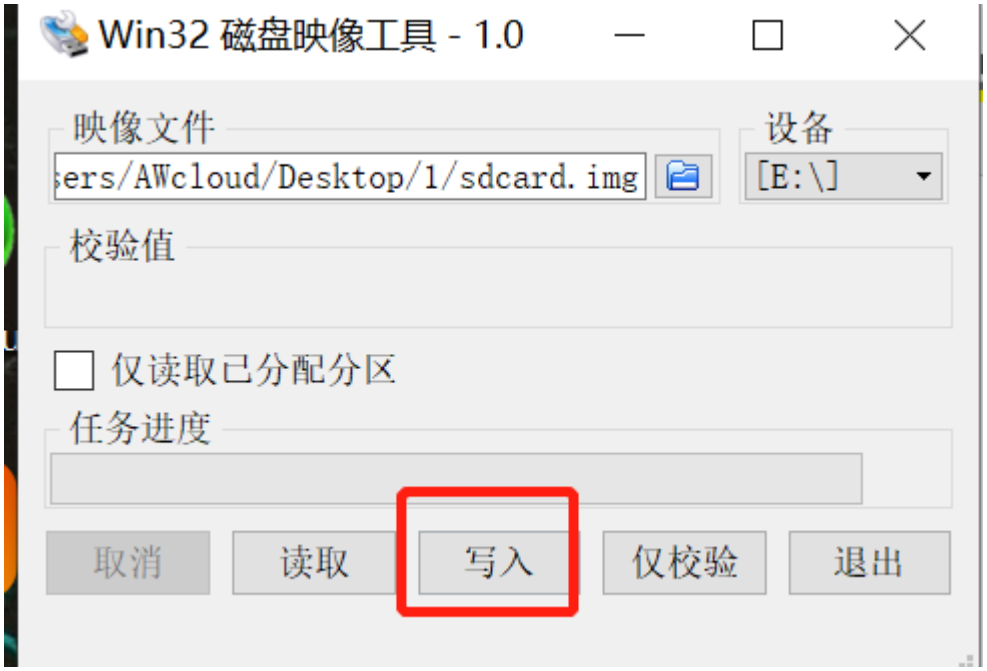
烧写镜像到 sd 卡
方式 1 ubuntu 烧写，这个方式 1M/s，要 10 多分钟
sudo dd if=sdcard.img of=/dev/sdb bs=2048
sync

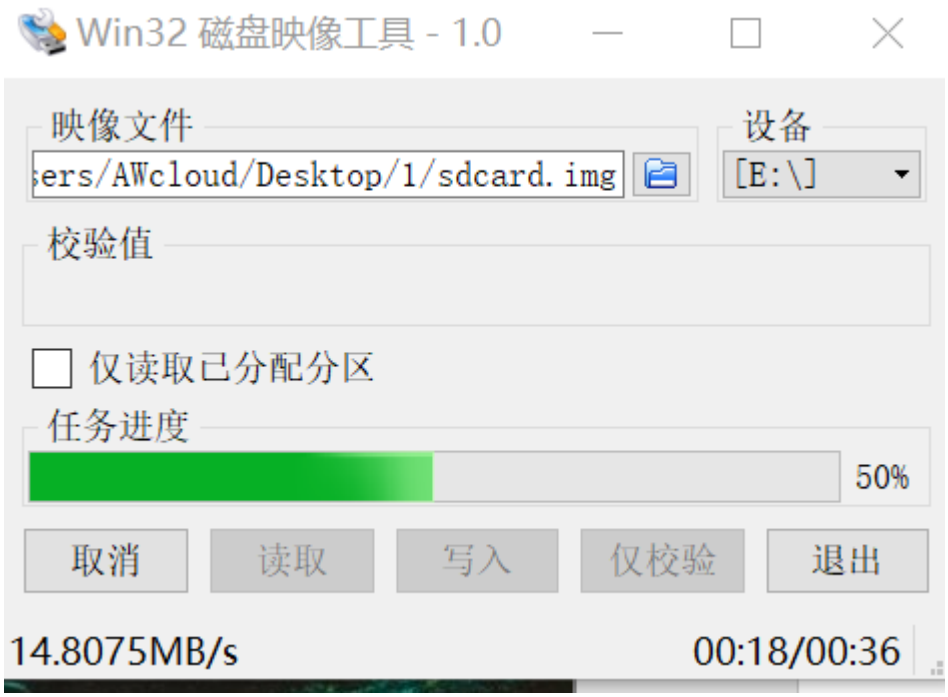
方式 2 将 sdcard.img 复制到 windows，用 win32disk 烧写
复制镜像到 windows



烧写，打开 win32disk，打开镜像


名称
修改日期
sdcard.img
2021/3/31 15:23
文件名(N):
sdcard.img
磁盘映像 (*.img *.IMG)
打开(O)
取消


Win32 磁盘映像工具 - 1.0
映像文件
ers/AWcloud/Desktop/1/sdcard.img
设备
[E:\]
校验值
仅读取已分配分区
任务进度
取消
读取
写入
仅校验
退出

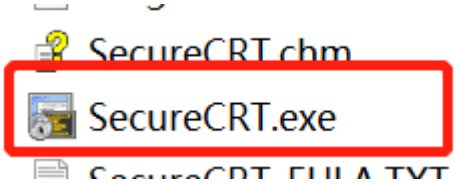


至此烧写完成

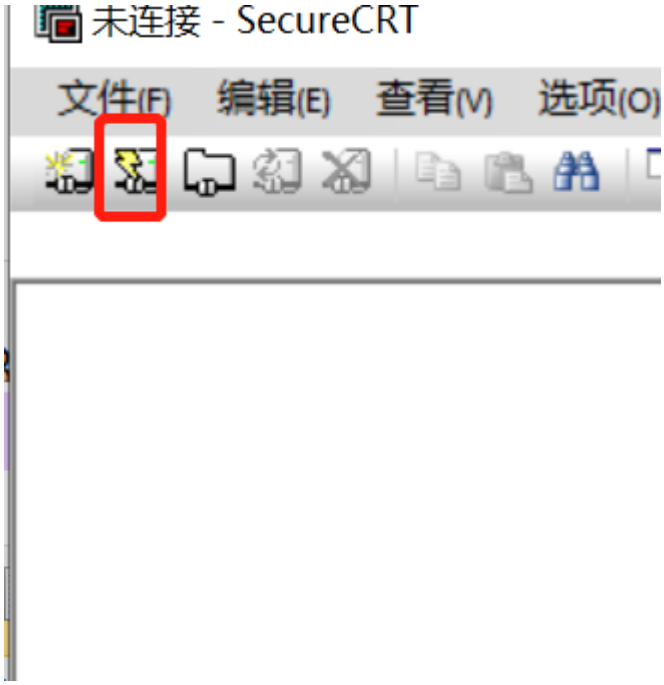
五、启动验证

板子连接好串口

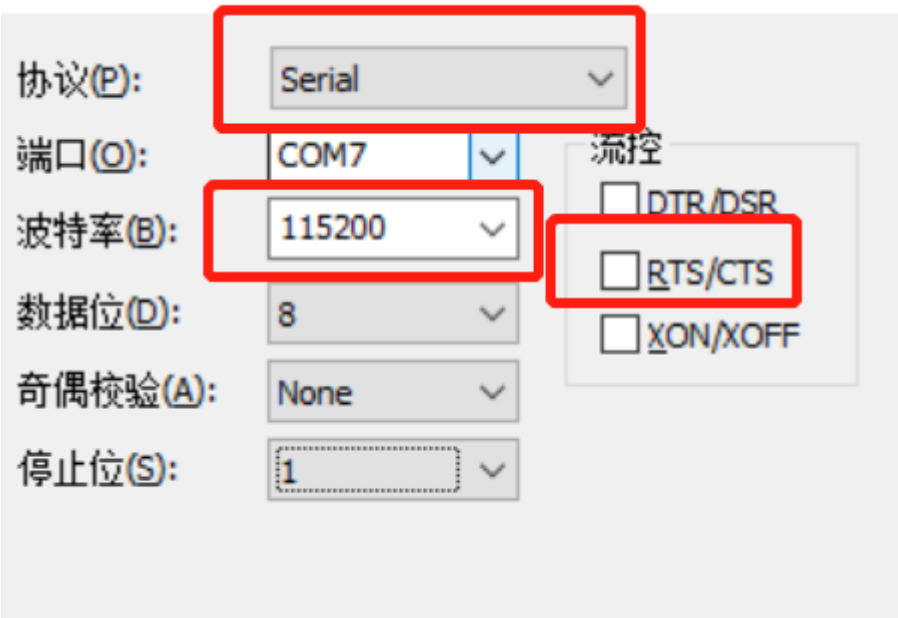
打开串口工具



配置信息如下



快速连接



开发板上电，查看终端信息

```
SEQ.C: CALIBRATION PASSED
SDRAM: 1024 MiB
ALTERA DWMMC: 0
reading u-boot.img
reading u-boot.img


U-Boot 2013.01.01 (Feb 05 2021 - 18:26:45)

CPU   : Altera SOCFPGA Platform
BOARD : Altera SOCFPGA Cyclone V Board
I2C:   ready
DRAM:  1 GiB
MMC:   ALTERA DWMMC: 0
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Skipped ethaddr assignment due to invalid EMAC address in EEPROM
Net:   mii0
Warning: failed to set MAC address

Hit any key to stop autoboot:  3
```

```
[    1.148255] of_cfs_init: OK
[    1.151575] ttyS0 - failed to request DMA
[    1.155774] Waiting for root device /dev/mmcblk0p2...
[    1.267793] mmc_host mmc0: Bus speed (slot 0) = 50000000Hz (slot req 50000000
Hz, actual 50000000HZ div = 0)
[    1.277560] mmc0: new high speed SDHC card at address aaaa
[    1.283486] mmcblk0: mmc0:aaaa SC16G 14.8 GiB
[    1.292184]  mmcblk0: p1 p2 p3
[    1.397644] EXT4-fs (mmcblk0p2): couldn't mount as ext3 due to feature incomp
atibilities
[    1.443527] random: fast init done
[    1.460853] EXT4-fs (mmcblk0p2): recovery complete
[    1.467019] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. O
pts: (null)
[    1.475125] VFS: Mounted root (ext4 filesystem) on device 179:2.
[    1.483852] devtmpfs: mounted
[    1.487782] Freeing unused kernel memory: 1024K
[    1.578968] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
Starting logging: OK
Initializing random number generator... done.
Starting network...

Welcome to Buildroot
guoguo login: 
```

默认网卡没有配置，需要自行配置才能上网

修改文件/etc/network/interfaces
vi /etc/network/interfaces
动态 ip 方式
添加
auto eth0
iface eth0 inet dhcp

```
Serial-COM7
# interface file auto-generated by buildroot

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

~
~
~
```

静态 ip 方式
添加
auto eth0
iface eth0 inet static
address 192.168.1.123
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameserve 114.114.114.114
保存

启动网卡（有时候一次不行，需要启动两次）
ifup eth0

```
# ifup eth0
udhcpc (v1.23.2) started
Sending discover...
Sending select for 172.16.64.242...
Lease of 172.16.64.242 obtained, lease time 86400
deleting routers
adding dns 218.201.17.2
adding dns 218.201.4.3
```

查看网络是否通畅 ping www.baidu,com

```
# ping www.baidu.com
PING www.baidu.com (183.232.231.172): 56 data bytes
64 bytes from 183.232.231.172: seq=0 ttl=55 time=29.428 ms
64 bytes from 183.232.231.172: seq=1 ttl=55 time=29.130 ms
64 bytes from 183.232.231.172: seq=2 ttl=55 time=28.546 ms
64 bytes from 183.232.231.172: seq=3 ttl=55 time=28.752 ms
64 bytes from 183.232.231.172: seq=4 ttl=55 time=28.534 ms
```

修改 ssh 连不上问题

修改 ssh 配置文件

vi /etc/ssh/sshd_config

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#RSAAuthentication yes
#PubkeyAuthentication yes

# The default is to check both .ssh/authoriz
# but this is overridden so installations wi
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none
```

保存退出，重启就可以了，eclipse 连接也不会提示密码错误了

至此，linux 搭建完成