IAC-IMX6UL Linux User Manual

Version: 1.0

2016.10.10

QIYANG INTELLIGENT TECHNOLOGY Co., Ltd

Catalogue

Preface

Product Description

Welcome to use IAC-I.MX6UL-KIT from Zhejiang Qiyang Intelligent Technology Co.,Ltd. It includes six manuals in Linux OS:

IAC-IMX6UL-KIT User Manual.PDF

IAC-IMX6UL-KIT Development Board Brif Introduction.PDF

IAC-IMX6UL-KIT BSP Development Instruction.PDF

IAC-IMX6UL-KIT Testing Manaul.PDF

IAC-IMX6UL-KIT Hardware Manual.PDF

IAC-IMX6UL-KIT Firmware Burning Manual.PDF

This manual mainly introduce that how to set up cross-compiler environment, source code and application example compiling .

Please read ***IAC-IMX6UL-KIT Hardware Manual.PDF*** carefully!

Target Readership

This manual is suitable for the following engineers:

- Testing Engineer
- Technical Support Engineer
- Software Engineer

Version Description

This manual is suitable for the following version:

Product Name	Version
IAC-IMX6UL-Kit	V1.0

Revision Record

Revision Record has accumulated the revised documentations description. The latest version has included the previous documentations updated contents.

Revision Date	Version	Revision Description

2016/10/10	V1.0	Launched

I. Illustration Before Using

- Built-in Linux System(ubuntu or other Linux distribution version), this manual uses ubuntu12.04, please refer to ***Ubuntu Installation for Virtual Machine Manual.PDF***
- Create working directory :[mkdir ~/work①②] in ubuntu user directory , copy the file to this directory which need to be complied.
- Here, we will not introduce the common commands and vi operation in Linux system. Please refer to the relative materials by yourself.
- The development board carry with DVD, all of the tools software and code file are in the corresponding directories. Please ensure all materials are there before use.
    - The DVD includes the ported source code, the user could compile directly. Or they can configure and compile as actual demanded.

Remark:

① Here, ~/ means the user directory,the command means that the actual route of creating work is /home/lvmh in this user directory.（This directory is account work directory, the lvmh account is the example in this manual.In the actual operation, please depend the actual account.）, the corresponding route of ~/work is /home/lvmh/work。

② For understanding and operating more convenient, all the files are copied in this directory, the actual user could create the actual directory. Here is just a example of ~/work!

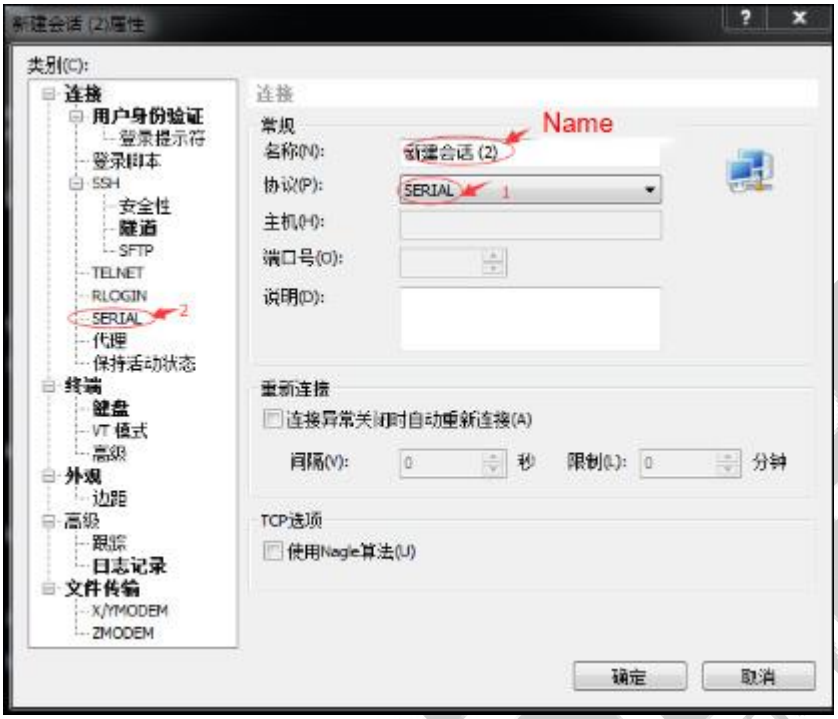
II. Preparation

- Serial Connection: Connect the J14 on development board with Serial Port on PC by the provided serial cable.
- Network connection: Connect (J24 and J3) Ethernet interface on development board to network interface on PC by LAN cable.

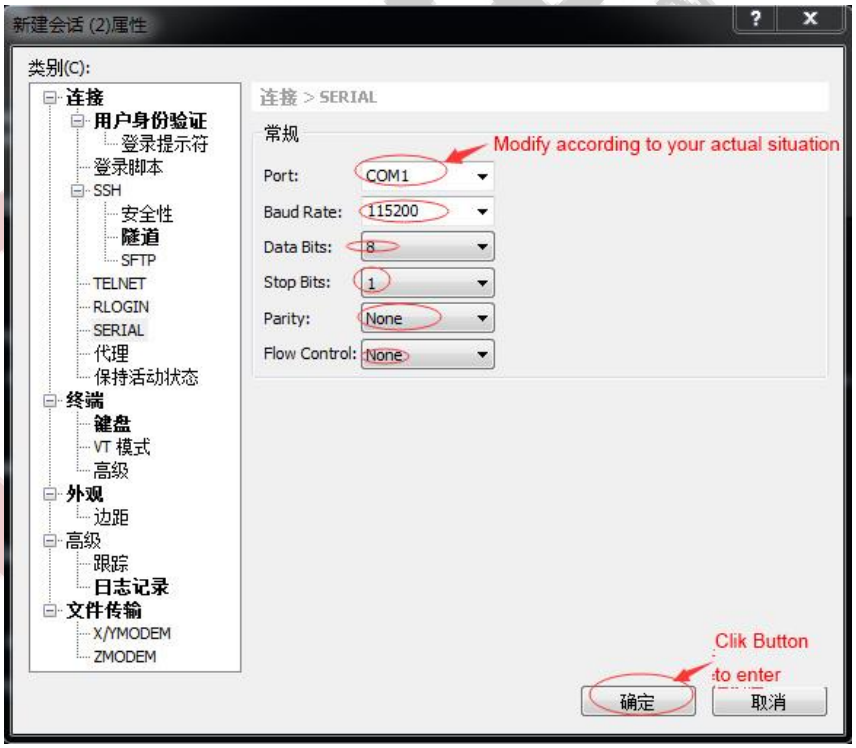
  ● USB connect: Connect (J6) USB Device on development board to USB on PC by USB cable.

  ● UART Setup: open terminal communication software Xshell (MiniCom or Windows hyperterminal), choose the needed serial ports and set up the parameters as below: Baud rate(115200), data bits(8 bits), stop bit(1 bit), check bit(none), data flow control(none), the details, please check the Picture 1-1,Picture 2-2:

Picture 1-1



Picture 1-2



Picture 1-3

III. Function Test

The procedures are used for testing whether the development board is in good functions, the development board pair with file system which integrated the testing program. You can find out relevant test program in [/usr/test] directory. The specific testing method , please refer to : ***IAC-IMX6UL-KIT Testing Manual.PDF***

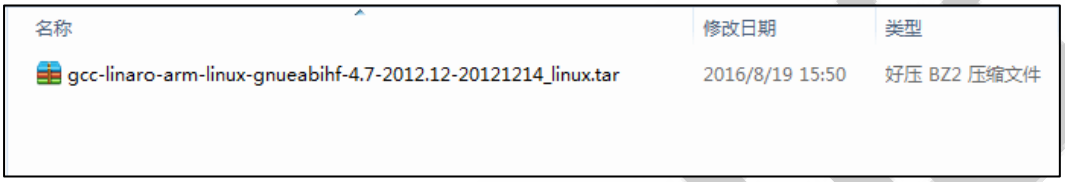
IV. Firmware Program

If need to re-load Linux firmware , the development board provides EMMC bootup method. The specific program method, please refer to ***IAC-IMX6UL-KIT Firmware Burning Manual.PDF***

V. Install Cross-compiler

5.1 Overview

Bootloader, kernel, fs, application programs and libraries all needs the cross-compiler. So, install the cross-compiler tool chain at first , there is already cross -compiler tool in DVD compiling tool. The user could install directly, the cross-compiler version GCC-4.7. As shown:



5.2 Installation Steps

Enter into [~/work] directory, copy[gcc-linaro-arm-linux-gnueabihf-4.7-2012.12-20 1214_linux.tar.bz2] to the following directory.

Steps:

Step 1:

Command	**1、$sudo tar –xjvf gcc-linaro-arm-linux-gnueabihf-4.7-2012.12-2012 14_linux.tar.bz2** **2、$ ls**
Description	Extract the cross-compiler and list the current file in current directory, to check whether it be extracted successfully.
Others	None
Reference	

```
lvmh@ubuntu:~/work$ ls
gcc-linaro-arm-linux-gnueabihf-4.7-2012.12-20121214_linux
gcc-linaro-arm-linux-gnueabihf-4.7-2012.12-20121214_linux.tar.bz2
```

Step 2:

Command	**$ vi ~/.bashrc**
Description	Add the path of cross compiler into the PATH system environment
Others	Add PATH① at the end of sentence
Reference	

```
export PATH=/home/lvmh/work/gcc-linaro-arm-linux-gnueabihf-4.7-2012.12-20121214_l
inux/bin:$PATH
```

Step 3:

Command	**$ source ~/.bashrc**
Description	Make the new environment variables effective
Others	None
Reference	None

Remark：

① export PATH=/home/lvmh/work/ gcc-linaro-arm-linux-gnueabihf-4.7-2012.12-201214_linux/bin:$PATH。

Step 4:

Command	**$ arm-linux-gnueabihf-gcc –v**
Description	Check the cross-compiler whether it is installed successfully.
Others	None

Reference

```
lvmh@ubuntu:~/work$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/home/lvmh/work/gcc-linaro-arm-linux-gnueabihf-4.7-2012.12-20
121214_linux/bin/../libexec/gcc/arm-linux-gnueabihf/4.7.3/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-lin
ux/.build/src/gcc-linaro-4.7-2012.12/configure --build=i686-build_pc-linux-gnu --
host=i686-build_pc-linux-gnu --target=arm-linux-gnueabihf --prefix=/cbuild/slaves
/oorts/crosstool-ng/builds/arm-linux-gnueabihf-linux/install --with-sysroot=/cbui
ld/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-linux/install/arm-linux-g
nueabihf/libc --enable-languages=c,c++,fortran --enable-multilib --with-arch=armv
7-a --with-tune=cortex-a9 --with-fpu=vfpv3-d16 --with-float=hard --with-pkgversio
n='crosstool-NG linaro-1.13.1-4.7-2012.12-20121214 - Linaro GCC 2012.12' --with-b
ugurl=https://bugs.launchpad.net/gcc-linaro --enable-__cxa_atexit --enable-libmud
flap --enable-libgomp --enable-libssp --with-gmp=/cbuild/slaves/oorts/crosstool-n
g/builds/arm-linux-gnueabihf-linux/.build/arm-linux-gnueabihf/build/static --with
-mpfr=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-linux/.build/a
rm-linux-gnueabihf/build/static --with-mpc=/cbuild/slaves/oorts/crosstool-ng/buil
ds/arm-linux-gnueabihf-linux/.build/arm-linux-gnueabihf/build/static --with-ppl=/
cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-linux/.build/arm-linu
x-gnueabihf/build/static --with-cloog=/cbuild/slaves/oorts/crosstool-ng/builds/ar
m-linux-gnueabihf-linux/.build/arm-linux-gnueabihf/build/static --with-libelf=/cb
uild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-linux/.build/arm-linux-
gnueabihf/build/static --with-host-libstdcxx='-L/cbuild/slaves/oorts/crosstool-ng
/builds/arm-linux-gnueabihf-linux/.build/arm-linux-gnueabihf/build/static/lib -lp
wl' --enable-threads=posix --disable-libstdcxx-pch --enable-linker-build-id --ena
```

VI. Set UP NFS Network File System

6.1 Overview

NFS is the abbreviation of Network File System. It could let the different mainframes share the files for development board use through NFS service. Take mainframes as the NFS server, then we can visit the files in mainframes through development board.

6.2 Installation Steps

Specific Steps:

Step 1:

Command	**$ sudo apt-get install nfs-kernel**
Description	Download and install
Others	None
Reference	None

Step 2:

Command	**$ sudo vi /etc/exports**
Description	Modify the end of [/etc/exports],create shared directory.
Others	Add ① at the end of the sentence.
Reference	

Remark:

① / nfs *(rw,sync,no_subtree_check, no_root_squash)。

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtr
ee_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/nfs *(rw,sync,no_root_squash,no_subtree_check)
~
```

Step 3:

Command	**$ sudo service nfs-kernel-server start**
Description	Start NFS service, [nfs-kernel-server] service
Others	None
Reference	

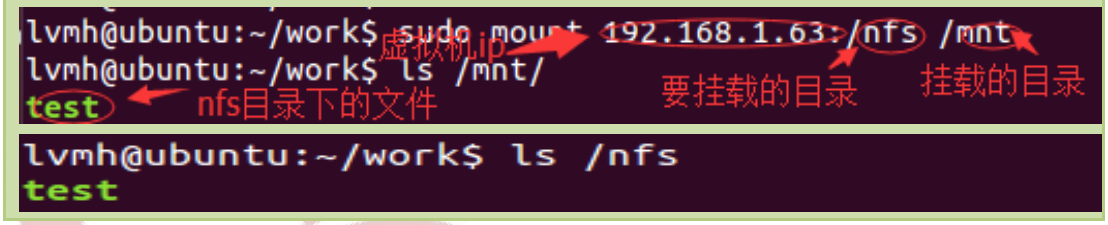
```
lvmh@ubuntu:~/work$ sudo service nfs-kernel-server start
 * Exporting directories for NFS kernel daemon...                    [ OK ]
 * Starting NFS kernel daemon                                        [ OK ]
```

Step 4:

Command	**$ sudo service portmap start**
Description	Start NFS server, boot portmap service
Others	None
Reference	

```
lvmh@ubuntu:~/work$ sudo service portmap start
portmap start/running, process 3453
```

Step 5:

Command	**1、$ sudo mount 192.168.1.63①:/ /mnt** **2、$ ls /mnt**
Description	Mount the mainframes to the root directory to check whether it be mounted successfully ②
Others	None
Reference	

Note：

If the development board cant mount to the host machine , you can try the following method to restart NFS server:

sudo exportfs -a

sudo /etc/init.d/portmap restart

sudo /etc/init.d/nfs-kernel-server restart

Remark:
① Use the actual virtual machine IP address.
② If you can see the contents from the root directory in [/mnt]directory, it means NFS configuration is correct.

VII. Source Code Compiling

7.1 Overview

Source code includes [uboot], [kernel], [file system]. Including, [uboot] and [kernel] needs compiling, file system needs to be created. This charter mainly introduce how to compile [uboot] and [kernel]. How to create the file system, please refer to Charter 9.

7.2 uboot Compiling

Copy [u-boot] source code to [~/work] working directory (The source code file name will be different base on the source code name and version.)

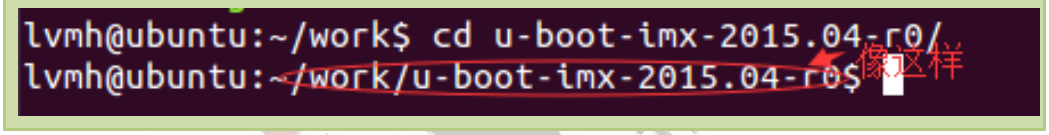
Specific Steps:

Step 1:

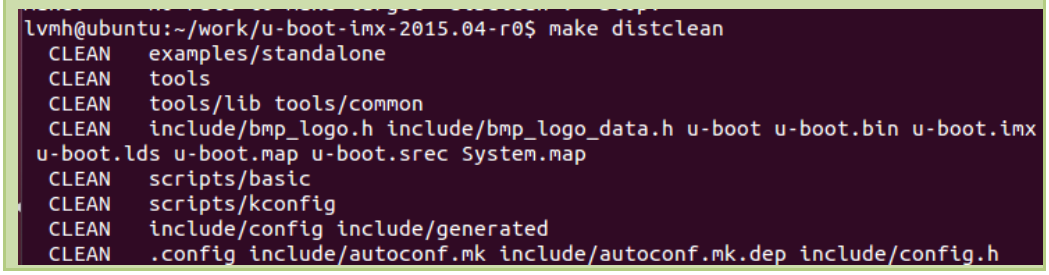
Command	**1、$ tar -xzvf uboot.tar.gz** **2、$ls**
Description	Extract the source code, and check whether it is extracted successfully
Others	None

Reference


Step 2:

Command	**$ cd u-boot-imx-2015.04-r0**
Description	Enter into source code directory
Others	None
Reference	

Step 3:

Command	**$ make distclean**
Description	Clean the generated file
Others	None
Reference	

Step 4:

Command	**$ make mx6ul_qiyang_defconfig**①
Description	Configure the board
Others	None
Reference	

```
lvmh@ubuntu:~/work/u-boot-imx-2015.04-r0$ make distclean
  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   include/config include/generated
  CLEAN   .config .config.old include/autoconf.mk include/autoconf.mk.dep include
/config.h
lvmh@ubuntu:~/work/u-boot-imx-2015.04-r0$ make mx6ul_qiyang_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

Step 5:

Command	**$ make –j8**②

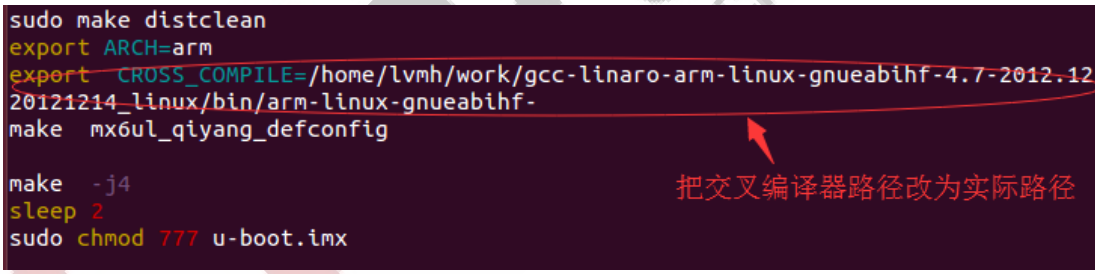
Remark:
① Configure uboot command [-make <board_name>_deconfig], the current board name is [mx6ul_qiyang].
② After execution, start compiling, it needs 1-3 minutes. The terminal will print some data. The last line means it could generate uboot image.

Description	Compile
Others	None
Reference	

Remark：

If compile unsuccessfully, please try this way:

1、$vi build.sh



After modification:

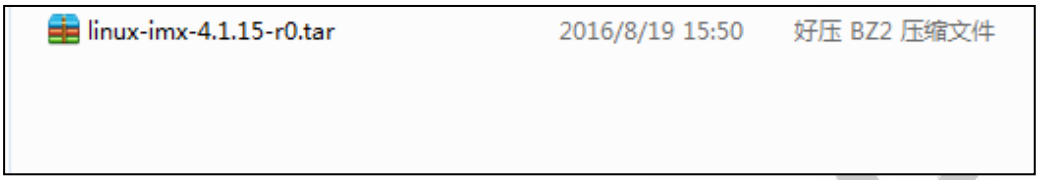
$ ./build.sh

2、$ vi Makefile

After modification, repeat uboot compiling steps.

7.3 Kernel configuration and compilation

Enter into [~/work]directory, copy [linux-imx-4.1.15-r0.tar.bz2] kernel source code to this directory (This source code name will be different base on the source code name and version updates].



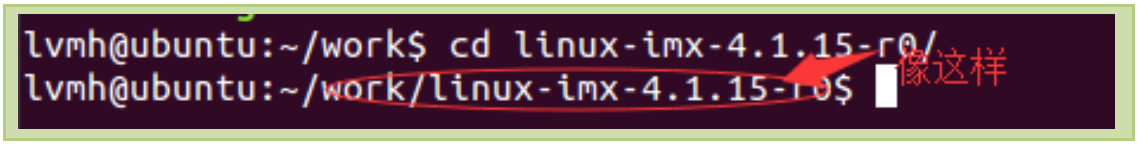
Compile Steps:

Step 1:

Command	**1、$ sudo tar –xjvf linux-imx-4.1.15-r0.tar.bz2** **2、$ ls**
Description	Exacted kernel source code
Others	None
Reference	

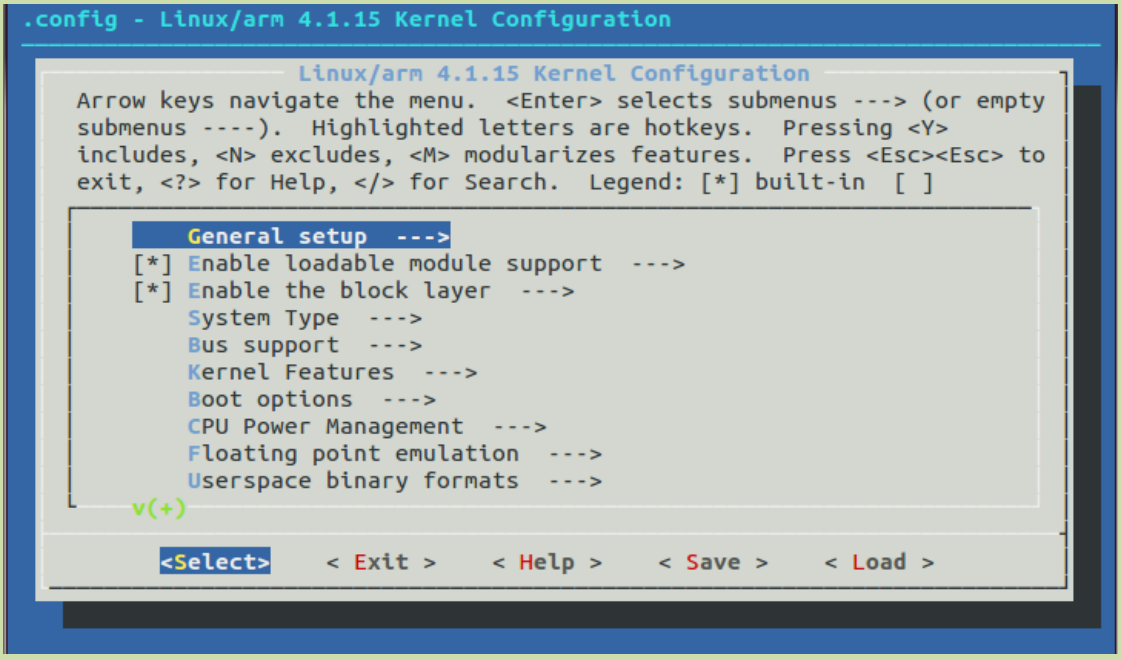
Step 2:

Command	**$ cd**
Description	Enter into kernel source code directory
Others	None
Reference	



Steps 3:

Command	**$ make menuconfig**①
Description	Configure Kernel
Others	None
Reference	



Steps 4:

Command	**$ make zImage dtbs**②
Description	Compile kernel and device tree.

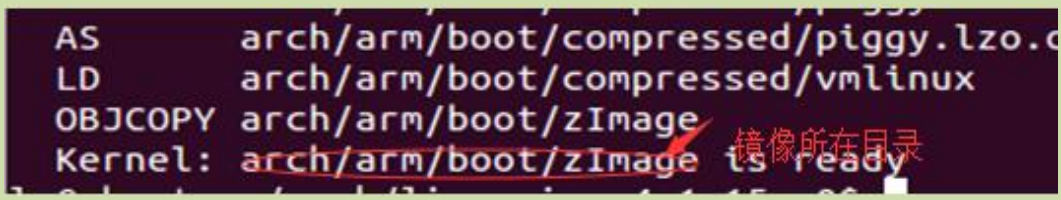
Remark：

① After executing, will show the kernel configuration interface as picture. Users will configure kernel,save and exit. After exit, please choose “YES”to save the configure.

②make zimage is the compile kernel source code.make dtbs is the compile kernel device tree.

Others	None

Reference

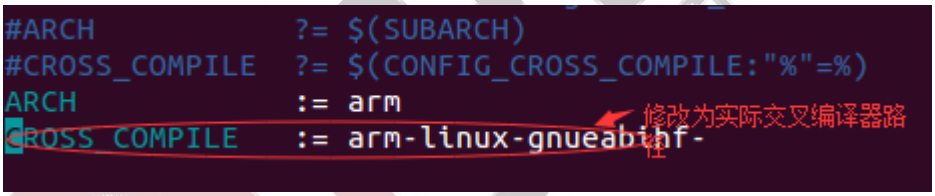


After compilation, it generates kernel image file [zImage] in [arch/arm/boot/] directory. And it generates device tree image file [imx6ul-qiyang.dtb] in [arch/arm/boot/dts] directory.

Remark：

If compile unsuccessfully, please do the following operations:

1、**$ vi Makefile**



Re-compile after modification

VIII. Application Program Development

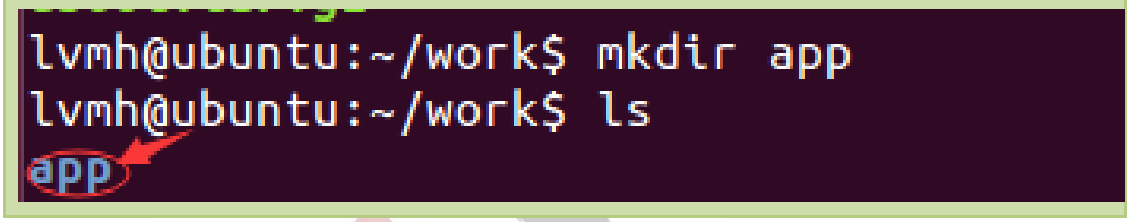
8.1 Compile application program and cross-compiler

Develop application program in [Linux], for example [Hello World]. Enter into [~/work] directory.

Specific Steps:

Step 1:

Command	**$ mkdir app**
Description	Create app folder
Others	None
Reference	



Step 2:

Command	**$ cd app**
Description	Enter into app folder
Others	None
Reference	

Step 3:

Command	**$ vi hello.c**①
Description	Compile application program
Others	None
Reference	

```
#include <stdio.h>
int main(void)
{
printf("Hello World ! \n");
return 0;
}
~
```

Step 4:

Command	**$ arm-linux-gnueabihf-gcc -o hello hello.c**② **$ ls**
Description	Compile application program ,then check whether it generated an executive file.
Others	None
Reference	

```
lvmh@ubuntu:~/work/app$ arm-linux-gnueabihf-gcc -o hello hello.c
lvmh@ubuntu:~/work/app$ ls
hello  hello.c
```
编译后得到的可执行程序

8.2 Run application program

We can get the executive program on development board. Now, we will introduce how the mainboard runs program on board. There are two methods: Mount NFS

Remark:
① After execution, show the blank page, then input programs in the page
② Need to use the pre-installed cross-compiler to do compiling work.

server and add to the file system[①]. This chapter mainly introduce the method to mount NFS server. How to let the virtual terminal software Xshell run ?

We can use the previous configured NFS server[②], Mount the NFS server on development board. Then you can operate the files from mainframe on development board. For example, copy file, run programs etc. These methods are also very good for debug.

Specific Steps:

Step 1:

Command	**$ ifconfig eth0 192.168.1.106**
Description	Set development board IP and sub network mask
Others	None
Reference	

```
root@qy_mx6ul:~# ifconfig eth0 192.168.1.106
root@qy_mx6ul:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:3E:26:0A:5B
          inet addr:192.168.1.106  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

自己设置的ip

Step 2

Command	**$ route add default gw 192.168.1.1**
Description	Set network
Others	None
Reference	

```
root@qy_mx6ul:~# route add default gw 192.168.1.1
```

① How to add ? Please refer to the Charter 6.
② Please ensure the network is connected well, NFS has been opened in PC.

Step 3:

Command	# **ping 192.168.1.63**
Description	PING virtual machine IP, test network connection.
Others	None
Reference	

```
root@qy_mx6ul:~# ping 192.168.1.63
PING 192.168.1.63 (192.168.1.63): 56 data bytes
64 bytes from 192.168.1.63: seq=0 ttl=64 time=2.226 ms
64 bytes from 192.168.1.63: seq=1 ttl=64 time=0.719 ms
64 bytes from 192.168.1.63: seq=2 ttl=64 time=0.704 ms
64 bytes from 192.168.1.63: seq=3 ttl=64 time=0.833 ms
64 bytes from 192.168.1.63: seq=4 ttl=64 time=0.716 ms
64 bytes from 192.168.1.63: seq=5 ttl=64 time=0.693 ms
64 bytes from 192.168.1.63: seq=6 ttl=64 time=0.728 ms
64 bytes from 192.168.1.63: seq=7 ttl=64 time=0.718 ms
64 bytes from 192.168.1.63: seq=8 ttl=64 time=0.749 ms
64 bytes from 192.168.1.63: seq=9 ttl=64 time=0.747 ms
64 bytes from 192.168.1.63: seq=10 ttl=64 time=0.722 ms

--- 192.168.1.63 ping statistics ---
11 packets transmitted, 11 packets received, 0% packet loss
round-trip min/avg/max = 0.693/0.868/2.226 ms
```

Step 4:

Command	# **mount –o nolock 192.168.1.63①:/ /mnt**
Description	Mount mainframe NFS server
Others	None

Remark:
① 192.168.1.63 is the virtual machine's IP.

Reference
root@qy_mx6ul:~# mount -o nolock 192.168.1.63:/nfs /mnt

Step 5:

Command	# **cd /mnt**
Description	Check whether it be mounted successfully
Others	None
Reference	root@qy_mx6ul:/mnt# ls test nfs目录下的文件

Step 6:

Command	$ **sudo cp hello /**
Description	Copy [hello] to root directory, the location [hello.c]on ubuntu.
Others	None
Reference	lvmh@ubuntu:~/work/app$ sudo cp hello /nfs lvmh@ubuntu:~/work/app$ ls /nfs/ hello test 复制过去的文件

Step 7:

Command	# **./hello**
Description	Run the useful program
Others	None
Reference	root@qy_mx6ul:/mnt# ./hello hello world! 应用程序打印信息

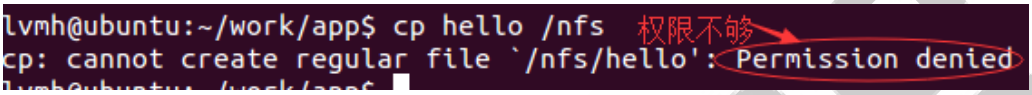
Through this way, it will be easy for debugging program. After the program be

debugged well, then add the application program to the file system, program to EMMC. It can avoid continual burning the program to the EMMC.

Remark:

If it shows the below mistake, please add [sudo] before the command:

$ sudo cp hello /



IX. Add Application Program to File System

9.1 Overview

Considering for easy production in mass production, we will packed the compiled application program ,libraries and configuration files to the file system.

9.2 Add to File System

Enter into [~/work] directory, copy the file system to the directory(The source code name will be different based on the source code name and version.)

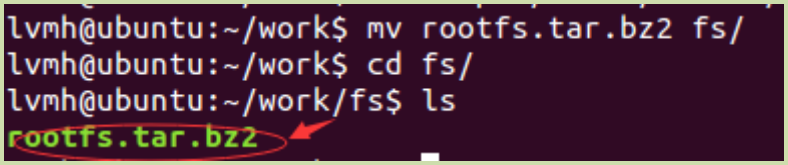
Adding Steps:

Step 1:

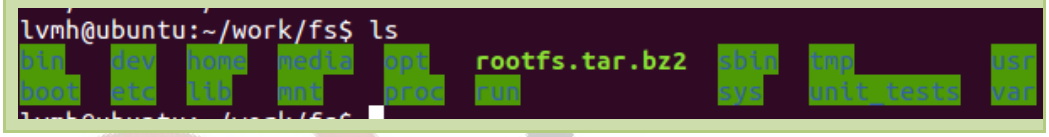
Command	**$ mkdir fs**
Description	Create fs folder
Others	None
Reference	lvmh@ubuntu:~/work$ mkdir fs lvmh@ubuntu:~/work$ ls app fs

Step 2:

Command	**$ mv rootfs.tar.bz2 fs/** **$ cd fs** **$ ls**
Description	Move file system source code [rootfs.tar.bz2]to [fs] folder. Enter into [fs]folder to check whether it be moved

	successfully.
Others	None
Reference	

Step 3:

Command	**1、$ sudo tar -xjvf rootfs.tar.bz2**①② **2、$ ls**
Description	Extract rootfs source code, then check whether it be extracted successfully
Others	None
Reference	

Step 4:

Command	**1、$ rm rootfs.tar.bz2** **2、$ ls**
Description	Delete the previous file system [rootfs.tar.bz2], to check whether it has been deleted.
Others	None
Reference	

Remark:
① The source code of file system extraction needs root permission. If you are general users, please add[sudo] before extraction
② Add each application program in each directory, library and configuration file.

```
lvmh@ubuntu:~/work/fs$ sudo rm rootfs.tar.bz2
lvmh@ubuntu:~/work/fs$ ls
bin   dev   home  media  opt   run   sys  unit_tests  var
boot  etc   lib   mnt    proc  sbin  tmp  usr
```

Step 5:

Command	**1、$ sudo tar -jcvf rootfs.tar.bz2 * -R** **2、$ ls**
Description	Re-compress file system
Others	None

Reference	`lvmh@ubuntu:~/work/fs$ ls`
	`bin  dev  home  media  opt  rootfs.tar.bz2  sbin  tmp  usr`
	`boot  etc  lib  mnt  proc  run  sys  unit_tests  var`

X. Conclusion

Thanks for using IAC-I.MX6UL-KIT.

If you have more questions, please contact our technical support : supports@qiyangtech.com