Compte rendu (20/5/2019 - 20/6/2019)

Etudiant : Yingjun HE

objective : quand on a réussi de suivre les procedures de la https://bootlin.com/doc/training/linux-kernel/linux-kernel-labs.pdf (lab de bootlin) on essaie de reformuler des procédure et utilise les sources officiels pour etre independant de la fichier qui founir par bootlin

sommaire :

1. les connaisant de git

telecharge git :
sudo apt install git gitk git-email
configuration global :
git config --global user.name 'My Name'
git config --global user.email me@mydomain.net

2. telecharge u-boot version ? et les sources

- git clone git://git.denx.de/u-boot.git
- git checkout v2018.05
- export CROSS_COMPILE=arm-linux-gnueabi-
- make am335x_boneblack_defconfig
- make

- wget https://bootlin.com/doc/training/linux-kernel/linux-kernel-labs.tar.xz
- tar xvf linux-kernel-labs.tar.xz

3. preparation compilation

sudo apt install gcc-arm-linux-gnueabi
(Install packages needed for configuring, compiling and booting the kernel for your board)
sudo apt install libssl-dev bison flex

4. compilation u-boot -> u-boot image et MLO

export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-

make def_config -> creer le fichier .config
make xconfige -> changer les option de la .config via interface graphique (procédure on utilise pas pour u-boot mais on va l’utiliser pour la kernel)
make -j 8 -> creer un image uboot et MLO

(commande exact)

- git clone git://git.denx.de/u-boot.git
- git checkout v2018.05
- export CROSS_COMPILE=arm-linux-gnueabi-
- make am335x_boneblack_defconfig

5. formater la micro SD (faire un micro SD bootable)

- installer le carte micro SD (directement ou bien avec adapateur)

vous allez trouver '/dev/mmcblk0' ou '/dev/sdb'

- puis faire la repartition la carte :
  sudo sfdisk /dev/mmcblk0 << EOF
  1,,0xE,*
  EOF

explication : separer apratire de 1Mb jusqua la fin. et puis ’* ‘ c’est à dire il va executer les fichier automatiquement (bootable)

- debancher et reinstaller la carte
- aussi pour être bootable il faut changer la format :

sudo mkfs.vfat -F 32 /dev/mmcblk0p1 -n boot (faire attention c’est fat 32)

- debancher et reinstaller la carte
- vous allez voir '/media/$USER/boot'

6. compiler zimage et dtb

- telecharge kernel : https://github.com/beagleboard (linux)
- git checkout -b 4.19 remotes/origin/4.19/linux-kernel-lab/src
- make bb.org_defconfig
- make xconfig (sudo apt install qt5-default) (pkg-config)
- déactiver la port usb-ethe :
    - CONFIG_USB_GADGET=n
    - CONFIG_USB_MUSB_HDRC=n
    - CONFIG_USB_MUSB_GADGET=n
    - CONFIG_USB_MUSB_DSPS=n
    - CONFIG_AM335X_PHY_USB=n
    - CONFIG_USB_ETH=n
    - CONFIG_PROVE_LOCKING =n
    - CONFIG_ROOT_NFS=y (permet de utiliser la NFS)
    - chercher la format compression XZ(general setup -> kernel compression) -> qui va generer un zImage sinon par defaut on va creer un Image
- make -j 8
- vous allez avoir zImage et am335x-boneblack.dtb (/arch/arm/boot/dtbs)
- cp zImage/var/lib/tftpboot

7. copier MLO et uboot

- cp MLO u-boot.img /media/$USER/boot

8. boot sur la carte micro SD

- picocom -b 115200 /dev/ttyUSB0 (pour lire les information sur la carte)
- sudo adduser $USER dialout (pour la permision)
- logout et login
- appuyer la bouton 'user' et puis bouton 'power'

9. bootargs montage NFS -> Nfsroot

(u - boot)

- env default -f -a
- saveenv
- setenv ipaddr 192.168.0.100
- setenv serverip 192.168.0.1
- setenv ethaddr 12:34:56:ab:cd:ef
- saveenv
- reset the borad
- setenv bootargs root=/dev/nfs ip=192.168.0.100:::::eth0 nfsroot=192.168.0.1:/home/hachicha/linux-kernel-labs/modules/nfsroot,nfsvers=3 rw (en une seul ligne)

(terminal)

- sudo apt install nfs-kernel-server
- /etc/exports (ecrit : /home/<user>/linux-kernel-labs/modules/nfsroot 192.168.0.100(rw,no_root_squash,no_subtree_check) attention il faut bien verifier les nom de path)
- sudo /etc/init.d/nfs-kernel-server restart
- nmcli con add type ethernet ifname enx??? ip4 192.168.0.1/24 (sur terminal et chercher ifconfig pour trouver l'adresse ethernet enx???)

10. charger kernel dans RAM

(terminal)

- sudo apt install tftpd-hpa
- copier zImage et am335x-boneblack.dtb à /var/lib/tftpboot

(u boot)

- tftp 0x81000000 zImage
- tftp 0x82000000 <board>.dtb
- bootz 0x81000000 - 0x82000000

11. comprendre le script qui permet de flasher la emmc

12. automatiser le boot processe

- setenv bootcmd 'tftp 0x81000000 zImage; tftp 0x82000000 am335x-boneblack.dtb; bootz 0x81000000 - 0x82000000'

13. busybox

- telecharge busybox 1.29 stable
- make defconfig
- make gconfig (il faut telecharge gconfig)
- build option -> build static binary
- destiantion path for 'make install' -> ../nfsroot
- make
- make install
- cd ../nfsroot
- mkdir dev
- mkdir proc sys root
- cd etc
- vim inittab ()

```
# Startup the system
null::sysinit:/bin/mount -t proc proc /proc
null::sysinit:/bin/mount -o remount,rw / # REMOUNT_ROOTFS_RW
null::sysinit:/bin/mkdir -p /dev/pts
null::sysinit:/bin/mkdir -p /dev/shm
null::sysinit:/bin/mount -a
null::sysinit:/bin/hostname -F /etc/hostname
# now run any rc scripts
::sysinit:/etc/init.d/rcS

# Put a getty on the serial port
# As CONFIG_SERIAL_8250_OMAP_TTYO_FIXUP=y by default,
# devtmpfs shows a /dev/ttyS0 device
ttyS0::respawn:/sbin/getty -L ttyS0 115200 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
null::shutdown:/etc/init.d/rcK
null::shutdown:/bin/umount -a -r
null::shutdown:/sbin/swapoff -a
```

- vim hostname
  ```
  bulidroot (comme vous voulez)
  ```
- vim passwd (configuration de password)
  ```
  root:x:0:0:root:/root:/bin/sh
  ```
- vim shadow (password codé, notre exempls est ‘vide’)
  ```
  root::10933:0:99999:7:::
  ```

- je n’ai pas trouver le compte et mot de passe par defaut donc j’ai besoin de passwd et shadow

14. share lib

- cp /usr/arm-linux-..../lib ~/nfsroot

15. site web

- cp /www ~/nfsroot
- /usr/sbin/httpd -h /www/

16. Third party libraries and applica-tions

- re-configer le noyeu
    - CONFIG_USB_GADGET=n
    - CONFIG_USB_MUSB_HDRC=n
    - CONFIG_USB_MUSB_GADGET=n
    - CONFIG_USB_MUSB_DSPS=n
    - CONFIG_AM335X_PHY_USB=n
    - CONFIG_USB_ETH=n
    - CONFIG_PROVE_LOCKING =n
    - CONFIG_ROOT_NFS=y (permet de utiliser la NFS)
    - chercher la format compression XZ(general setup -> kernel compression) -> qui va generer un zImage sinon par defaut on va creer un Image
    - CONFIG_SOUND =y
    - CONFIG_SND=y
    - CONFIG_SND_USB=y
    - CONFIG_SND_USB_AUDIO=y
- make -j 8
- vous allez avoir zImage et am335x-boneblack.dtb (/arch/arm/boot/dtbs)
- cp zImage/var/lib/tftpboot
- cd ~/embedded-linux-labs/thirdparty
- mkdir staging target (creer un staging place et targe place)
- sudo cp -a ~/nfsroot/* target/
- setenv bootargs root=/dev/nfs ip=192.168.0.100:::::eth0 nfsroot=192.168.0.1:/home/hachicha/embedded-linux-labs/thirdparty/target,nfsvers=3 rw
- /home/hachicha/embedded-linux-labs/thirdparty/target 192.168.0.100(rw,no_root_squash,no_subtree_check)
- telecharge alsa-lib 1.1.6
- CC=arm-linux-gnueabi-gcc ./configure --host=arm-linux-gnueabi
- make
- cd src/.lib
- arm-linux-gnueabi-gcc -shared conf.o confmisc.o input.o output.o \async.o error.o dlmisc.o socket.o shmarea.o \userfile.o names.o -lm -ldl -lpthread -lrt \-Wl,-soname -Wl,libasound.so.2 -o libasound.so.2.0.0
- ln -s libasound.so.2.0.0 libasound.so.2
- ln -s libasound.so.2.0.0 libasound.so
- make clean(inutil les etape avant cest pour comprendre)
- CC=arm-linux-gnueabi-gcc ./configure --host=arm-linux-gnueabi --disable-python --prefix=/usr
- make
- make DESTDIR=$HOME/embedded-linux-labs/thirdparty/staging install
- mkdir ~/targetusr/lib
- cd ~/embedded-linux-labs/thirdparty/staging/usr/lib
- cp -a libasound.so.2* ~/embedded-linux-labs/thirdparty/target/usr/lib

- arm-linux-gnueabi-strip target/usr/lib/libasound.so.2.0.0 (strip the library)
- telecharge alsa-utilis 1.1.6
- CC=arm-linux-gnueabi-gcc ./configure --host=arm-linux-gnueabi --prefix=/usr
- LDFLAGS=-L$HOME/embedded-linux-labs/thirdparty/staging/usr/lib \CPPFLAGS=-I$HOME/embedded-linux-labs/thirdparty/staging/usr/include \CC=arm-linux-gnueabi-gcc \./configure --host=arm-linux --prefix=/usr \--disable-alsamixer --disable-xmlto
- make DESTDIR=$HOME/embedded-linux-labs/thirdparty/staging/ install
- cd ..
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/bin/a* $HOME/embedded-linux-labs/thirdparty/staging/usr/bin/speaker-test $HOME/embedded-linux-labs/thirdparty/target/usr/bin/
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/sbin/alsa* $HOME/embedded-linux-labs/thirdparty/target/usr/sbin
- arm-linux-gnueabi-strip $HOME/embedded-linux-labs/thirdparty/target/usr/bin/a*
- arm-linux-gnueabi-strip $HOME/embedded-linux-labs/thirdparty/target/usr/bin/speaker-test
- arm-linux-gnueabi-strip $HOME/embedded-linux-labs/thirdparty/target/usr/sbin/alsactl
- mkdir -p $HOME/embedded-linux-labs/thirdparty/target/usr/share/alsa/pcm
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/share/alsa/alsa.conf* $HOME/embedded-linux-labs/thirdparty/target/usr/share/alsa/
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/share/alsa/cards $HOME/embedded-linux-labs/thirdparty/target/usr/share/alsa/
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/share/alsa/pcm/default.conf $HOME/embedded-linux-labs/thirdparty/target/usr/share/alsa/pcm/
- telecharge libogg 1.3.3
- CC=arm-linux-gnueabi-gcc ./configure --host=arm-linux-gnueabi --prefix=/usr
- make
- make DESTDIR=$HOME/embedded-linux-labs/thirdparty/staging/ install
- cd ..
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/lib/libogg.so.0* $HOME/embedded-linux-labs/thirdparty/target/usr/lib/
- arm-linux-gnueabi-strip $HOME/embedded-linux-labs/thirdparty/target/usr/lib/libogg.so.0.8.3
- telecharge libvorbis 1.3.6
- CC=arm-linux-gnueabi-gcc ./configure --host=arm-linux-gnueabi --prefix=/usr --with-ogg-includes=$HOME/embedded-linux-labs/thirdparty/staging/usr/include --with-ogg-libraries=$HOME/embedded-linux-labs/thirdparty/staging/usr/lib
- make
- make DESTDIR=$HOME/embedded-linux-labs/thirdparty/staging/ install
- cd ..
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/lib/libvorbis.so.0* $HOME/embedded-linux-labs/thirdparty/target/usr/lib/
- arm-linux-gnueabi-strip $HOME/embedded-linux-labs/thirdparty/target/usr/lib/libvorbis.so.0.4.8

- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/lib/libvorbisfile.so.3* $HOME/embedded-linux-labs/thirdparty/target/usr/lib/
- arm-linux-gnueabi-strip target/usr/lib/libvorbisfile.so.3.3.7
- telecharge libao 1.2.0
- LDFLAGS=-L$HOME/embedded-linux-labs/thirdparty/staging/usr/lib \CPPFLAGS=-I$HOME/embedded-linux-labs/thirdparty/staging/usr/include \CC=arm-linux-gnueabi-gcc ./configure --host=arm-linux \--prefix=/usr
- make
- make DESTDIR=$HOME/embedded-linux-labs/thirdparty/staging/ install
- cd ..
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/lib/libao.so.4* target/usr/lib/
- arm-linux-gnueabi-strip $HOME/embedded-linux-labs/thirdparty/target/usr/lib/libao.so.4.1.0
- mkdir -p $HOME/embedded-linux-labs/thirdparty/target/usr/lib/ao/plugins-4/
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/lib/ao/plugins-4/libalsa.so $HOME/embedded-linux-labs/thirdparty/target/usr/lib/ao/plugins-4/
- telecharge vorbis-tools 1.4.0
- LDFLAGS=-L$HOME/embedded-linux-labs/thirdparty/staging/usr/lib \CPPFLAGS=-I$HOME/embedded-linux-labs/thirdparty/staging/usr/include \PKG_CONFIG_PATH=$HOME/embedded-linux-labs/thirdparty/staging/usr/lib/pkgconfig \PKG_CONFIG_SYSROOT_DIR=$HOME/embedded-linux-labs/thirdparty/staging \LIBS=-lm \CC=arm-linux-gnueabi-gcc \./configure --host=arm-linux-gnueabi --prefix=/usr --without-curl
- make
- make DESTDIR=$HOME/embedded-linux-labs/thirdparty/staging/ install
- cd ..
- cp -a $HOME/embedded-linux-labs/thirdparty/staging/usr/bin/ogg* $HOME/embedded-linux-labs/thirdparty/target/usr/bin
- arm-linux-gnueabi-strip $HOME/embedded-linux-labs/thirdparty/target/usr/bin/ogg*
- arm-linux-gnueabi-strip $HOME/embedded-linux-labs/thirdparty/target/lib/*

17. Using a build system, example withBuildroot

- telecharge buildroot 2019.02
- mkdir buildroot
- tar vxf buildroot-2019.02.3.tar.gz
- cd buildroot-2019.02.3
- make menuconfig
- Target options
    - Target Architecture:ARM (little endian)
    - Target Architecture Variant:cortex-A8 (lire les doc)
    - Enable VFP extension support: Enabled
    - Target ABI:EABIhf
    - Floating point strategy:VFPv3-D16
- Toolchain
    - Toolchain type:External toolchain
    - Toolchain:Custom toolchain
    - Toolchain path: use the toolchain you built:/home/<user>/x-tools/arm-training-linux-uclibcgnueabihf
    - External toolchain gcc version:8.x
    - External toolchain kernel headers series:4.16.x
    - External toolchain C library:uClibc/uClibc-ng
    - We must tell Buildroot about our toolchain configuration, so selectToolchain hasWCHAR support?,Toolchain has SSP support?andToolchain has C++ support?.Buildroot will check these parameters anyway.
- Target packages
    - KeepBusyBox(default version) and keep the Busybox configuration proposed byBuildroot;
    - Audio and video applications
        - Selectalsa-utils
        - ALSA utils selection
            - Selectalsactl
            - Selectalsamixer
            - Selectspeaker-test
        - Selectvorbis-tools
    - Filesystem images
        - Selecttar the root filesystem
- make
- cd $HOME/embedded-linux-labs/buildroot/
- mkdir nfsroot
- sudo tar xvf ../buildroot-2019.02/output/images/rootfs.tar

- sudo vim /etc/exports
    - /home/hachicha/embedded-linux-labs/buildroot/nfsroot 192.168.0.100(rw,no_root_squash,no_subtree_check)
- sudo /etc/init.d/nfs-kernel-server restart
- (u boot) setenv bootargs root=/dev/nfs ip=192.168.0.100:::::eth0 nfsroot=192.168.0.1:/home/hachicha/embedded-linux-labs/buildroot/nfsroot,nfsvers=3 rw
- il faut changer dans inittab
    - ttyS0::respawn:/sbin/getty -L ttyS0 115200 vt100 # GENERIC_SERIAL
- ogg123 or speaker-test
-

18. Application development

- cd $HOME/embedded-linux-labs/appdev
- export PATH=$HOME/embedded-linux-labs/buildroot/buildroot-XXXX.YY/output/host/usr/bin:$PATH
- arm-linux-gcc -o app app.c $(pkg-config --libs --cflags ncurses)

19.

-

explication :
MBR : pour charger u-boot https://fr.wikipedia.org/wiki/Master_boot_record
LOG LEVEL : kernel hacking printk and dmesg options
beagle bone :
http://www.bootembedded.com/embedded-linux/building-embedded-linux-scratch-beaglebone-black/

http://www.bootembedded.com/beagle-bone-black/building-custom-root-file-system-rfs-for-beagle-bone-black-using-busy-box/

uclibc : https://blog.csdn.net/u011011827/article/details/62237087
mount : https://blog.csdn.net/ylyuanlu/article/details/24555945