

Partie II – Compilation du noyau

Christophe BLAESS

Mai 2010

La petite carte IGEPv2 disponible depuis quelques mois chez ISEE (voir <http://www.igep-platform.com/>) est une plateforme performante, économique et très complète pour s'initier au développement Linux embarqué. Il existe déjà plusieurs distributions spécifiques pour cette carte, mais nous étudions dans cette série d'article comment construire son propre système en partant de zéro..."

Dans le premier article de cette série, nous avons mis en place une chaîne de compilation croisée (*cross-toolchain*) basé sur GCC qui nous permet de générer du code pour le processeur de la carte IGEPv2. Nous allons à présent passer à une seconde phase, très importante : la compilation du noyau Linux.

Préparation des sources

La première étape consiste à télécharger les sources et les décompresser. Nous allons utiliser un noyau récent, car le support pour la carte IGEP v2 y a été ajouté depuis peu.

```
$ cd ~/Projet_Igep
$ wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.33.tar.bz2
2
[...]
2010-05-09 23:56:54 (1,08 MB/s) - «linux-2.6.33.tar.bz2» sauvegardé
[66266488/66266488]
$ tar -xjf linux-2.6.33.tar.bz2
$
```

Nous allons également avoir besoin de l'utilitaire *mkimage* issu du projet *U-boot*. Suivant les distributions, vous pouvez le trouver sous forme de *package* précompilé, ou avoir besoin de le compiler vous-même. Auquel cas, il faut télécharger :

```
$ wget ftp://ftp.denx.de/pub/u-boot/u-boot-2010.03.tar.bz2
2010-05-10 00:03:34 (765 KB/s) - «u-boot-2010.03.tar.bz2» sauvegardé
[8844729]
$ tar -xjf u-boot-2010.03.tar.bz2
$
```

Préparation de *mkimage*

Si vous disposez déjà de l'utilitaire *mkimage* de *U-boot*, vous pouvez passer tout de suite au paragraphe suivant. Sinon nous allons compiler une petite partie de *U-boot*, juste pour obtenir cet utilitaire.

```
$ cd u-boot-2010.03
$ make omap3_beagle_config
```

Configuring for omap3_beagle board...

Comme vous le voyez, nous configurons *U-boot* pour la carte *BeagleBoard*. Ceci est nécessaire car il n'existe pas encore de pré-configuration pour IGEPv2, et les deux cartes sont très proches.

```
$ export PATH=$PATH:~/cross/arm-linux/usr/bin/
$ make
[...]
u-boot-2010.03/lib_arm/bootm.c:122: undefined
reference to 'udc_disconnect'
make: *** [u-boot] Erreur 1
```

La compilation s'arrête en erreur, mais cela ne nous dérange pas, l'utilitaire *mkimage* a déjà été généré dans le répertoire *tools*. Il suffit de le copier dans un répertoire du *PATH*.

```
$ cp tools/mkimage ~/bin
$ cd ..
```

Configuration du noyau

Entrons dans le répertoire des sources de Linux :

```
$ cd linux-2.6.33/
```

Un fichier de configuration existe pour notre carte. Nous devons le copier dans notre répertoire sous le nom *.config* (le point initial dans le nom du fichier ne nous permet pas de le voir avec *ls* sans l'option *-a*).

```
$ cp arch/arm/configs/igep0020_defconfig ./config
```

Puis nous allons examiner la configuration du noyau en exécutant :

```
$ make ARCH=arm menuconfig
```

Attention : il est important de respecter la casse (majuscule/minuscule) de l'argument *ARCH=arm*.

Parcourez quelques instants l'interface de configuration de la compilation du noyau, vous trouverez de nombreux points intéressants. En autres, dans le menu « *General Setup* », sur la seconde ligne, l'option « *Local version* » vous permet de préciser un suffixe qui sera ajouté après le numéro du noyau. Ce suffixe (par exemple vos initiales) vous permettra d'identifier votre noyau personnalisé.

Compilation du noyau

```
$ make ARCH=arm CROSS_COMPILE=~/cross/arm-linux/usr/bin/arm-linux-
[...]
IHEX firmware/kaweth/trigger_code_fix.bin
$
```

La variable *CROSS_COMPILE* est un préfixe qui est systématiquement ajouté devant les commandes de compilation (*gcc*, *ar*, *as*). Ainsi la chaîne de compilation du noyau

invocera-t-elle « `~/cross/arm-linux/usr/bin/arm-linux-gcc` » au lieu de « `gcc` ».

La compilation va durer quelques minutes. Une fois celle-ci terminée, reprenez votre ligne et ajoutez « `uImage` » à la fin. Ceci va permettre d'obtenir (à l'aide de `mkimage`) un fichier acceptable par le *bootloader U-boot*.

```
$ make ARCH=arm CROSS_COMPILE=~/cross/arm-linux/usr/bin/arm-linux-  
uImage  
[...]  
Data Size:      1816616 Bytes = 1774.04 kB = 1.73 MB  
Load Address:  80008000  
Entry Point:   80008000  
Image arch/arm/boot/uImage is ready  
$
```

L'image du noyau prête à être chargée par le *bootloader* est donc disponible. Maintenant, il nous reste à choisir une méthode pour lui fournir cette image. Il y en a essentiellement trois :

- Mettre l'image dans un répertoire d'un serveur *TFTP*. Ceci est assez contraignant car par défaut la carte *IGEPv2* contacte un serveur à l'adresse IP 192.168.254.10, et recherche les images dans le répertoire `poky/poky-image-sato/igep0020b/` du serveur. Bien sûr ces éléments sont paramétrables en modifiant la configuration du *bootloader* pré-installé.
- Transférer l'image dans la mémoire flash centrale en utilisant le lien série de débogage et en dialoguant avec le *bootloader*. Plus facile à mettre en oeuvre, mais légèrement plus risqué car une erreur peut conduire à l'effacement de l'image initiale.
- Placer l'image dans une partition de la carte flash micro-SD que l'on peut rajouter sur la carte *IGEPv2*. Ceci est simple et moins dangereux que de toucher à la mémoire flash interne.

Nous allons donc utiliser la troisième méthode.

Formatage de la carte micro-SD

Procurez vous une petite carte micro-SD de quelques gigaoctets. Nous allons la diviser en deux partitions. Dans la première partition nous déposerons l'image du noyau, et dans la seconde nous construirons le système de fichiers.

Nous formaterons la première partition avec le format *VFat*. Ceci permettra à *Uboot* de lire la partition, et permettra de l'utiliser éventuellement pour échanger des données avec d'autres systèmes d'exploitation.

Insérez la carte micro-SD (via un adaptateur USB par exemple) dans votre système hôte (celui sur lequel vous préparez l'image). Suivant votre système, la carte SD peut apparaître avec un nom différent de celui présenté ici. Généralement en insérant la carte vous verrez une fenêtre apparaître, vous demandant si vous voulez copier des photos, des fichiers, etc. Fermez cette fenêtre en notant le nom du périphérique affiché, et démontez la carte.

Utilisez `fdisk` pour créer deux partitions distinctes sur cette carte (en détruisant au besoin la partition initiale d'abord). Voici un exemple avec une carte de 1 Go que nous partitionnons en 300 Mo et 700 Mo (la carte apparaît ici sous le nom `/dev/mmcb1k0`).

Il est souvent nécessaire d'utiliser un compte *root* pour faire ces opérations ou d'utiliser *sudo*.

```
# fdisk /dev/mmcblk0
[...]
Command (m for help): p
Disk /dev/mmcblk0: 1019 MB, 1019215872 bytes
4 heads, 16 sectors/track, 31104 cylinders
Units = cylinders of 64 * 512 = 32768 bytes
Disk identifier: 0x15f136ac

    Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1          1         31104     995320   83   Linux

Command (m for help): d
Selected partition 1

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-31104, default 1): 1
Last cylinder, +cylinders or +size{K,M,G} (1-31104, default 31104):
300M
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (301-31104, default 301): (entrée)
Last cylinder, +cylinders or +size{K,M,G} (301-31104, default 31104):
Using default value 31104
Command (m for help): p
[...]

    Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1          1           300        9592    83   Linux
/dev/mmcblk0p2        301         31104     985728   83   Linux

Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
#
```

Formatons la première partition :

```
# mkfs.vfat /dev/mmcblk0p1
mkfs.vfat 3.0.0 (28 Sep 2008)
#
```

Puis copions-y l'image du noyau :

```
# mount /dev/mmcblk0p1 /mnt/
# cp /home/cpb/Projet_Igep/linux-2.6.33/arch/arm/boot/uImage /mnt/
# umount /mnt/
```

Et insérons la carte-SD dans le connecteur de la IGEPv2.

Boot du noyau

Avant de tester notre noyau, je vous conseille de relier le port série RS-232 de la carte IGEPv2 à votre PC afin de voir les messages de démarrage. Pour cela il vous faudra un petit montage *null-modem* dont on trouve facilement le schéma sur Internet.

En outre, la plupart d'entre vous utiliseront un adaptateur *USB-Série*, qui donne accès au port sur `/dev/ttyUSB0`.

La configuration du port est établie par *U-boot* à 115200 bits / sec, 8 bits de données et pas de parité.

Pour visualiser les messages du noyau, il est utile de lancer sur la machine hôte un émulateur de terminal, par exemple le fameux *minicom*.

Voici des extraits commentés d'une trace obtenue après avoir mis la carte IGEPv2 sous tension :

```
Texas Instruments X-Loader 1.4.2-1 (Mar 22 2010 - 08:57:12)
```

```
Detected Numonyx OneNAND 4G Flash  
Loading u-boot.bin from onenand
```

Le *bootloader* de Texas Instrument (*X-Loader*) trouve une image de *U-Boot* dans la mémoire *flash* interne et la charge.

```
U-Boot 2009.11-1 (Mar 22 2010 - 10:41:15)  
OMAP3530-GP ES3.1, CPU-OPP2 L3-165MHZ  
IGEP v2 board + LPDDR/ONENAND  
DRAM: 512 MB
```

U-boot a reconnu l'architecture pour laquelle il est compilé, et vérifie la mémoire flash interne.

```
Muxed OneNAND(DDP) 512MB 1.8V 16-bit (0x58)  
OneNAND version = 0x0031  
Chip support all block unlock  
Chip has 2 plane  
block = 2048, wp status = 0x2  
Scanning device for bad blocks  
[...]  
Hit any key to stop autoboot: 3
```

Pendant le décompte, il est possible de « prendre la main » sur U-boot pour modifier ses paramètres. Ici nous allons le laisser continuer avec ses options par défaut.

```
mmc0 is available  
reading boot.ini  
** Unable to read "boot.ini" from mmc 0:1 **  
reading uImage  
1816680 bytes read  
## Booting kernel from Legacy Image at 80000000 ...
```

| *U-boot* recherche boot.ini, puis se tourne vers uImage qu'il charge et à laquelle il transmet l'exécution.

```
[...]  
Starting kernel ...  
Uncompressing Linux... done, booting the kernel.  
[ 0.000000] Linux version 2.6.33-logilin (cpb@Inspiron) (gcc  
version 4.3.4 (Buildroot 2010.02) ) #1 Sun May 23 00:41:56 CEST 2010
```

| Dès le début du *boot*, le noyau affiche son numéro de version. Vous pouvez remarquer l'extension que nous avons ajouté au moment de la compilation.

```
[ 0.000000] CPU: ARMv7 Processor [411fc083] revision 3 (ARMv7),  
cr=10c53c7f  
[ 0.000000] CPU: VIPT nonaliasing data cache, VIPT nonaliasing  
instruction cache  
[ 0.000000] Machine: IGEP v2 board
```

| etc...

```
[ 10.343017] musb_hdrc: USB Host mode controller at fa0ab000 using  
DMA, IRQ 92  
[...]  
[ 10.343963] hub 1-0:1.0: USB hub found  
[...]  
[ 10.366119] Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
```

```
[...]  
[ 10.885345] console [ttyS2] enabled  
[...]  
[ 10.910186] smsc911x: Driver version 2008-10-21.  
[...]  
[ 11.433197] mmc0: new high speed SD card at address 1234  
[...]  
[ 11.536773] List of all partitions:  
[ 11.540313] b300          995328 mmcblk0 driver: mmcblk  
[ 11.545654] b301          9592 mmcblk0p1  
[ 11.549987] b302          985728 mmcblk0p2  
[ 11.554290] No filesystem could mount root, tried: ext3 ext2 vfat  
msdos  
[ 11.561096] Kernel panic - not syncing: VFS: Unable to mount root  
fs on unknown-block(179,2)
```

Notre noyau a parfaitement reconnu ses périphériques (USB, série, réseau, carte SD...), il a initialisé toutes ses tables internes, et essaye à présent de monter le périphérique *block* sur lequel *U-boot* lui a promis qu'il trouverait son système de fichiers. Il essaye donc de monter la partition 2 de la carte SD et échoue car elle n'est pas formatée.

Nous pouvons résoudre ce problème facilement, il faut éteindre la carte IGEPv2, récupérer la carte micro-SD et l'insérer à nouveau dans notre PC de développement. Puis formater (par exemple en *ext2*) la seconde partition.

```
# mkfs.ext2 /dev/mmcblk0p2  
mke2fs 1.41.4 (27-Jan-2009)  
Étiquette de système de fichiers=  
Type de système d'exploitation : Linux
```

```
[...]
Le système de fichiers sera automatiquement vérifié tous les
28 montages ou après 180 jours, selon la première éventualité
Utiliser tune2fs -c ou -i pour écraser la valeur.
#
```

Nous démontons la mémoire, la ré-insérons dans la carte IGEPv2 et redémarrons cette dernière. A présent le boot se termine ainsi :

```
[ 11.473510] mmc0: host does not support reading read-only switch.
assuming write-enable.
[ 11.481719] mmc0: new high speed SD card at address 1234
[ 11.487579] mmcblk0: mmc0:1234 SA01G 972 MiB
[ 11.492248] mmcblk0: p1 p2
[ 11.592224] VFS: Mounted root (ext2 filesystem) on device 179:2.
[...]
[ 11.608245] Kernel panic - not syncing: No init found. Try passing
init= option to kernel.
```

Le VFS (*Virtual File System*) du noyau a bien monté notre partition. Puis il cherche à lancer le premier processus du système, le fameux « init ». Naturellement, notre partition étant vide, il ne le trouve pas et échoue en un *kernel panic* familier des développeurs embarqués.

Cela peut vous paraître surprenant, mais lorsque je réalise des portages de Linux sur des cartes embarquées spécifiques, j'éprouve un grand soulagement à l'apparition de ce message *kernel panic* car il indique que le noyau a effectué correctement son premier *boot*.

Il nous reste donc à continuer notre construction en installant un processus *init*. Ceci fera l'objet de notre prochain article...

Christophe BLAESS

Retrouvez toutes ces manipulations dans notre formation
« Linux Embarqué et Temps-Réel »
sur <http://www.logilin.fr/>