

Développement Linux autour de la plate-forme ARM

Pierre Ficheux (pierre.ficheux@openwide.fr)

CTO Open Wide Ingénierie

Mars 2011

- Présentation d'Open Wide
- Historique ARM
- Bootloader
- Noyau Linux
- Ajout de support de carte ARM
- Outils de développement
 - Distributions
 - Buildroot, OE
 - émulateur QEMU

- SSII/SSL créée en septembre 2001 avec Thales et Schneider
- Indépendant depuis 2009
- Environ 90 salariés sur Paris et Lyon
- Industrialisation de composants open source
- Quatre activités :
 - OW : système d'information
 - OW outsourcing: hébergement
 - OW ingénierie: informatique industrielle
 - OW technologies: composants Java



- Fabriqué au départ par Acorn (UK) → ARM = Acorn Risc Machine (1983/85)
- ARM2 8MHz (1987) → Archimedes
- ARM3 (1989)
- Création de Advanced RISC Machine (Acorn, Apple, VLSI) → ARM6 → vente de licences ARM
- ARM7 (v3) / ARM7TDMI (v4) / ARM7EJ
- ARM8
- ARM9 !
- XScale
- ARM11 → multicoeur
- Cortex



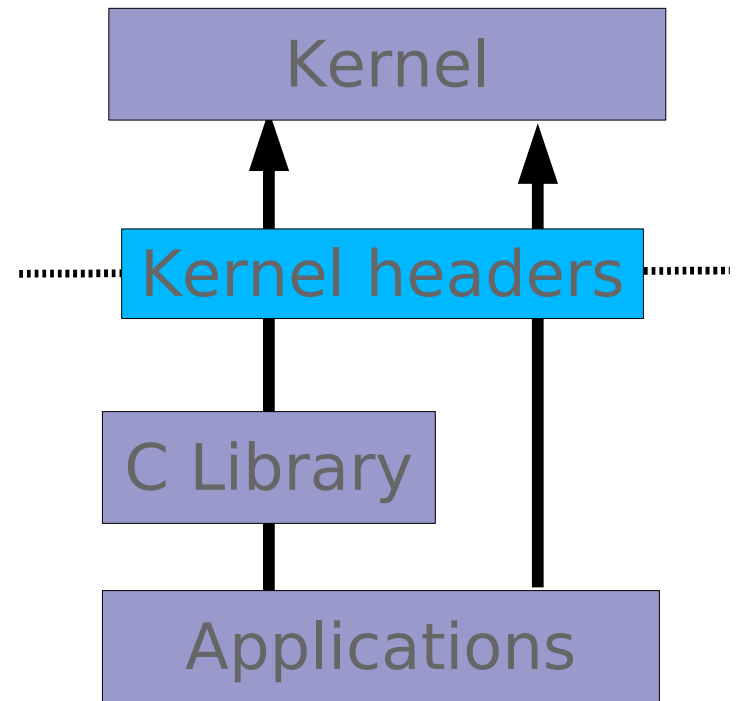
- Atmel AT91xx
 - Freescale i.MX !
 - SAMSUNG S3C24xx
 - NXP
 - Broadcom
 - Marvell
 - ST
 - TI
 - Apple
- Nombreux domaines d'application: téléphonie, IAD, industriel, ...

- Choix / génération d'une chaîne de compilation croisée
 - ELDK
 - Code Sourcery
 - Crosstool, Crosstool-NG
- Adaptation d'un bootloader :
 - U-Boot
 - BareBox (aka U-Boot v2)
 - RedBoot
- Adaptation du noyau Linux
- Le portage dépend de la carte et non pas uniquement du CPU (# x86)



- Un point *très* complexe, même si ce n'est pas spécifique à l'ARM
- Nécessité de construire une chaîne « croisée » :
 - GCC
 - Binutils (as, ld, ...)
 - Dépendances avec le noyau (*system calls, ...*)
=> erreur « Kernel too old »
 - Choix d'une libC => Glibc, uClibc, Eglibc, ...
 - GDB
 - Toute autre bibliothèque utilisée => libstdc++
+
 - Dépendances avec le compilateur hôte

- Interaction entre la libC et le noyau Linux
 - Appels systèmes (nombre, définition)
 - Constantes
 - Structures de données, etc.
- Compiler la libC – et certaines applications - nécessite les entête du noyau
- Disponibles dans `<linux/...>` et `<asm/...>` et d'autres répertoires des sources du noyau (`include, ...`)



- Numéro des *system calls*, dans `<asm/unistd.h>`

```
#define __NR_exit      1
#define __NR_fork     2
#define __NR_read     3
```

- Définition de constantes dans `<asm/generic-fcntl.h>`, inclus par `<asm/fcntl.h>`, inclus par `<linux/fcntl.h>`

```
#define O_RDWR        00000002
```

- Structures de données dans `<asm/stat.h>`

```
struct stat {
    unsigned long    st_dev;
    unsigned long    st_ino;
    [...]
};
```

- Utiliser un compilateur binaire :
 - ELDK: <http://www.denx.de/wiki/DULG/ELDK>
 - Code Sourcery:
<http://www.codesourcery.com/sgpp/lite/arm/portal/release644>
- Installation simple
- Support (payant) possible
- Configuration connue => support par les forums
- Par contre:
 - Version des composants figées
 - Choix libC limité

- Construire un compilateur
 - Crosstool => obsolète
 - Crosstool-NG => assez complexe à prendre en main
 - Buildroot / OpenEmbedded
- Aucun ou n'est « plug and play »
- La mise au point peut prendre des jours, voire plus !
- Outils produits : arm-linux-* (gcc, as, ld, ar, nm, ...)

- Maintenu par DENX (ELDK)
- U-Boot v2 (Barebox) maintenu par Pengutronix
- Très bon support ARM
- Adaptation à partir d'une carte existante
- arch/arm/cpu/ → choix du CPU
- board/ → liste des cartes par fabricant

```
$ ls board/arm\ltd
```

```
integrator versatile
```

```
$ make versatile_config
```

```
Configuring for versatile board...
```

```
Variant:: PB926EJ-S
```

```
$ make ARCH=arm CROSS_COMPILE=arm-linux
```

- Multi-plateforme (26 actuellement)
- Code dépendant de l'architecture dans arch/
- Pour ARM, 5% de code dépendant de l'architecture
- Développement de référence « orienté » x86
- Compilation ARM « in a nutshell »
 - \$ export ARCH=arm
 - \$ export CROSS_COMPILE=arm-linux
 - \$ make versatile_defconfig
 - \$ make uImage
- Résultat dans arch/arm/boot/uImage à utiliser avec U-Boot


- Pour certains modèles ARM :
 - application de « patch » externe (constructeur, communauté) → AT91 sur http://maxim.org.za/at91_26.html
 - noyau fourni par un fabricant de carte → problème de mise à jour car le noyau n'est pas « mainline »
 - Parfois plusieurs versions de noyau suivant la source
 - Problème de compatibilité avec des extensions noyau (PREEMPT-RT, Xenomai)
- Le problème n'est pas lié à ARM !

- Cas fréquent: adapter le noyau pour un carte proche d'un modèle supporté (carte d'évaluation)
- Pré-requis: CPU est déjà supporté !
 - Ajouter la définition de la carte dans le répertoire `arch/arm/mach-TYPE_CPU/`
 - `arch/arm/mach-at91/board-myboard.c`
 - Mettre à jour `Makefile` et `Kconfig`
 - Ajouter une configuration de référence à `arch/arm/configs`
 - Ajouter un identifiant numérique à `arch/arm/tools/mach-types` (pour U-Boot)

- Définition de la carte dans board-myboard.c :

```
MACHINE_START(AT91SAM9263EK, "Atmel AT91SAM9263-EK")
    /* Maintainer: Atmel */
    .phys_io          = AT91_BASE_SYS,
    .io_pg_offst      = (AT91_VA_BASE_SYS >> 18) &
0xffffc,
    .boot_params      = AT91_SDRAM_BASE + 0x100,
    .timer            = &at91sam926x_timer,
    .map_io           = ek_map_io,
    .init_irq         = ek_init_irq,
    .init_machine     = ek_board_init,
MACHINE_END
```

- Problèmes :
 - Cette méthode (MACHINE_START) est fréquente mais pas « normalisée » (voir mach-ns9xxx)
 - Le portage reste complexe
 - Pas mal de code dupliqué
- Solution → utilisation du « device tree »
 - Code générique pour l'initialisation, utilisant une définition de la carte dans un fichier .dts (source) / .dtb (compilé)
 - Utilisé sur PowerPC depuis 2005
 - Essentiel pour SoC (Xilinx, Altera)
 - Démarrage un peu plus lent...
 - En cours sur ARM !

- Il existe deux « ABI » pour ARM (Application Binary Interface)
 - OABI → Old ABI
 - EABI → Embedded ABI
- Meilleures performances en EABI (flottant)
- Les compilateurs récents produisent du code EABI
- Le noyau Linux *doit* être configuré pour le support EABI dans le menu *Kernel features/Use the ARM EABI to compile the kernel* 
- Dans le cas contraire, *crash* lors de l'exécution de *init*

- La plupart des distributions classiques sont orientées x86
- Quelques distributions existent pour ARM
 - Debian GNU/Linux
 - ArchMobile (Arch Linux)
 - Ubuntu (?)
- Pour l'embarqué, on utilise plutôt :
 - Distribution commerciale (WindRiver, MV)
 - Distribution fournie par le fabricant de carte (SDK, qualité inégale)
 - Outil de production (Buildroot, OE, OpenWrt)

- Indépendant de la cible → support bootloader + noyau
- Un « moteur » crée la distribution à partir des sources des composants adaptés en appliquant des « patch »
- Ne fournit pas les sources: uniquement les patch et les règles de production en tenant compte des dépendances :-)
- Peut produire la chaîne croisée
- Produit la distribution sous diverses formes
 - Image du bootloader
 - Noyau Linux (zImage, uImage)
 - Image de root-filesystem en EXT2, JFFS2, CRAMFS, TAR, CPIO, ...

- OpenEmbedded
 - Moteur écrit en Python
 - Très puissant mais lourd
 - Basé sur des fichiers de configuration
- Buildroot
 - Au départ un démonstrateur pour uClibc
 - Désormais un véritable outil, bien maintenu !
- OpenWrt
 - Dérivé de BR
 - Orienté vers les IAD (Internet Access Device)
- Autres: LTIB (Freescale), PTXdist (Pengutronix)

- Lié au projet uClibc (micro-C-libc) : libC plus légère que la Glibc
- But initial: produire des images de test (root-filesystem) de uClibc
- Moteur basé sur des fichiers *Makefile* et des scripts-shell
- Par défaut, utilise Busybox
- Outil de configuration utilise le langage *Kconfig*
- Produit également la chaîne de compilation (uniquement basée sur uClibc !)
- Pas de version « officielle » avant 2009
- Très actif aujourd'hui

- Télé-charger depuis <http://buildroot.uclibc.org>
- Extraire l'archive
- Configurer par `make menuconfig`
- Compiler par `make`
- Le résultat est dans le répertoire `output/images`
 - Bootloader (U-Boot)
 - Noyau Linux
 - Image(s) du root-filesystem
- La chaîne de compilation dans `output/staging`

- Une « généralisation » de l'approche utilisée dans BR
- Utilise un moteur écrit en Python (bitbake) et un ensemble de règles utilisant un principe d'héritage => « recipe » (recette)
- Pas d'interface de configuration
- Processus lourd => plusieurs heures pour la première compilation (environ 30mn pour BR)
- TRES puissant, recommandé dans le cas où l'on gère un grand nombre de configurations
- Gère la notion de paquet binaire, contrairement à BR

- QEMU est un émulateur de matériel open source
- Initialement développé par Fabrice Bellard et diffusé sous GPL v2
- Disponible pour les principales distributions
- Utilisation :
 - Emulation de cartes ARM pour test
 - Développement si matériel manquant
 - Compatibilité binaire pour les programmes en espace utilisateur
 - Emulation du framebuffer par QEMU → applications graphique (Qt, ...)

- Installer QEMU sur le poste de développement →
qemu-system-arm
- 23 cartes ARM supportées

```
$ qemu-system-arm -M ?
```

```
Supported machines are:
```

```
syborg      Syborg (Symbian Virtual Platform)
```

```
musicpal    Marvell 88w8618 / MusicPal (ARM926EJ-S)
```

```
...
```

```
versatilepb ARM Versatile/PB (ARM926EJ-S)
```

```
versatileab ARM Versatile/AB (ARM926EJ-S)
```

```
integratorcp ARM Integrator/CP (ARM926EJ-S) (default)
```

- Produire le noyau + la distribution (avec BR)
- Utiliser les deux fichiers zImage et rootfs.cpio.gz
- Emulation d'une console série (RS-232) ou FB
- Syntaxe à utiliser
 - \$ qemu-system-arm -M versatilepb -m 64 -kernel zImage -initrd rootfs.gz -append "console=ttyAMA0,115200" -nographic
- Options
 - -append : passe les options au noyau
 - -nographic : pas de SDL => émulation de la console série
 - -M : type de carte émulée
 - -m : mémoire allouée en Mo

← Console série

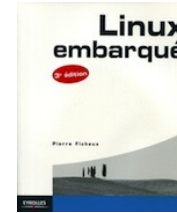
- Test en RAM avec l'image u-boot.bin :

```
# qemu-system-arm -M versatilepb -m 64  
-kernel u-boot.bin -nographic -net  
nic,macaddr=52:54:00:12:34:xx -net  
tap
```

- La configuration n'est pas sauvegardée car nous travaillons en RAM :- (→ utiliser une flash !
- L'émulation de la flash (NOR) est possible sous-réserve de quelques adaptations dans QEMU et U-Boot.

```
# qemu-system-arm -M versatilepb -m 64  
-pflash flash.img -nographic ...
```

- Actuellement ARM est LA plateforme embarquée universelle
- Favorisée par la téléphonie et IAD (nouvelle Freebox)
- Attaquée par Intel avec Atom (exemple: STB) mais d'autres architectures sont plus vulnérables (MIPS, SH4, ...)
- Très bon support Linux (2ème après x86 ?) mais complexe en l'absence de normalisation (device tree)
- Très bon support QEMU permettant de se former à ARM sans matériel :)



- Linux embarqué version 3 :
<http://www.eyrolles.com/Informatique/Livre/linux-embarque-9782212124521>
- linux@arm.linux.org.uk (Liste ARM Linux)
- http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MCIMX51EVKJ
- <http://www.arm.linux.org.uk>
- <http://www.linuxembedded.fr/2010/11/larrivee-des-flattened-device-tree>
- <http://www.linuxuk.org/docs/elce2009/elc2009-device-trees-for-arm>
- <http://www.denx.de/wiki/U-Boot>
- <http://thomas.enix.org/Blog-20081002153859-Technologie>
- <http://free-electrons.com/doc/toolchains.odp>
- <http://www.pragmatec.net>
- <http://www.eukrea.com>

