

# Buildroot, un outil de développement pour Linux embarqué

Pierre Ficheux ([pierre.ficheux@openwide.fr](mailto:pierre.ficheux@openwide.fr))

CTO Open Wide / OS4I

21/04/2010

- Si un équipement utilise un OS Linux embarqué, on doit produire les différents éléments :
  - Le bootloader (exemple: U-Boot)
  - L'image du noyau Linux « statique » (zImage)
  - L'image du « root-filesystem » contenant les commandes Linux standards, les modules, les applications liées à la fonction de l'équipement
- L'équipement n'est pas forcément du x86 !

# Création d'une distribution LE

- Utiliser un produit d'éditeur (Wind River, MV, ...) => €€€
- Adapter une distribution Linux classique
  - Limité au niveau matériel
  - Empreinte mémoire importante
- Créer la distribution « from scratch »
  - Compétences nécessaires
  - Difficile à industrialiser: gestion des dépendances et des évolutions
- Utiliser un outil de génération : Buildroot, OpenEmbedded, LTIB

# Principes des outils de génération

- 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 prenant en compte les dépendances :-)
- Fournit la chaîne croisée: « GNU Toolchain » (gcc, gas, gdb...)
- Produit la distribution sous diverses formes
  - Image du bootloader
  - Noyau Linux (zImage, uImage)
  - Image de root-filesystem en EXT2, JFFS2, CRAMFS, TAR, CPIO, ...



# Quelque outils disponibles

- OpenEmbedded
  - Moteur écrit en Python
  - Très puissant mais lourd
  - Basé sur des fichiers de configuration
- LTIB
  - Utilisé surtout par Freescale
  - Interface de configuration similaire à celle du noyau
- Buildroot
  - Au départ un démonstrateur pour uClibc
  - Désormais un véritable outil, bien maintenu !



# Origine de Buildroot

- Lié au projet uClibc (micro-C-libc) : libc + légère et portable que la Glibc
- But initial: produire des images de test 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, uniquement sur SVN

# Buildroot aujourd'hui

- Repris en 2009 par Peter Korsgaard et Thomas Petazzoni
- Une version officielle tous les 3 mois: 2009.02, ..., 2009.11, 2010.02
- Projet géré sous Git
- Documentation améliorée
- Plus de 300 composants adaptés
- Support CPU x86, ARM, PowerPC, SH4, ...
- Support de cartes Atmel, ...
- Il est assez « simple » d'ajouter un support de carte...si l'on connaît bien le Makefile et Kconfig



## Using Buildroot « in a nutshell »

- 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
  - Noyau Linux
  - Image(s) du root-filesystem
- Nécessite une connexion Internet pour télécharger les archives !

# Configuration de Buildroot

- Type de CPU + variante, ex: arm puis arm920t
- Options de la cible (très simple si pré-définie)
- Choix de la chaîne de compilation: interne (uClibc) ou externe (Glibc, Eglibc, ...)
- Composants de la distribution (packages)
- Formats des images du root-filesystem
  - JFFS2, CRAMFS, SQUASHFS (flash)
  - EXT2, CPIO (ramdisk)
  - TAR (NFS Root)
- Noyau Linux et bootloader (non compilés par défaut)



- Avec QEMU:
  - `$ qemu-system-arm -M versatilepb -m 64 -kernel output/images/zImage -initrd output/images/rootfs.arm.cpio`
- Sur carte réelle
  - En utilisant TFTP pour charger le noyau + NFS-Root pour le root-filesystem
  - Flash du noyau + image rootfs en utilisant le bootloader (nand write...)
  - Eventuellement, on flashe le bootloader avec JTAG ou port série
- Voir la démonstration BR / QEMU !

# Avantages de Buildroot

- Stable, maintenu par des experts européens de Linux embarqué
- Interface commune à plusieurs cartes: facilité d'utilisation, de maintenance, de sous-traitance
- Configurateur graphique par « make xconfig »
- Léger et rapide: on produit le compilateur puis la distribution en 20/25 mn sur un PC classique, quelques minutes si la chaîne est externe
- Devient une référence dans le monde embarqué

- Moins « puissant » qu'OpenEmbedded, pas de notion de « recette » (classes) mais OE est très lourd et ne dispose pas d'interface de configuration
- Pour l'instant pas de gestion de paquetages binaires (IPKG/OPKGS) contrairement à OpenEmbedded ou OpenWrt
- Gère uniquement UNE partition pour le root-filesystem

# Questions?

