open wide
INGENIERIE

Linux embarqué, matériel et outils

Pierre Ficheux (pierre.ficheux@openwide.fr)

Décembre 2011

open wide
INGENIERIE

Programme

- Matériel compatible
- Composants logiciels d'une distribution « Linux embarqué »
- Outils de développement et de mise au point (GNU Toolchain, QEMU)
- Outils de production de distributions (Buildroot, OpenEmbedded)
- Offre commerciale / outils libres et « gratuits » (avantages / inconvénients)

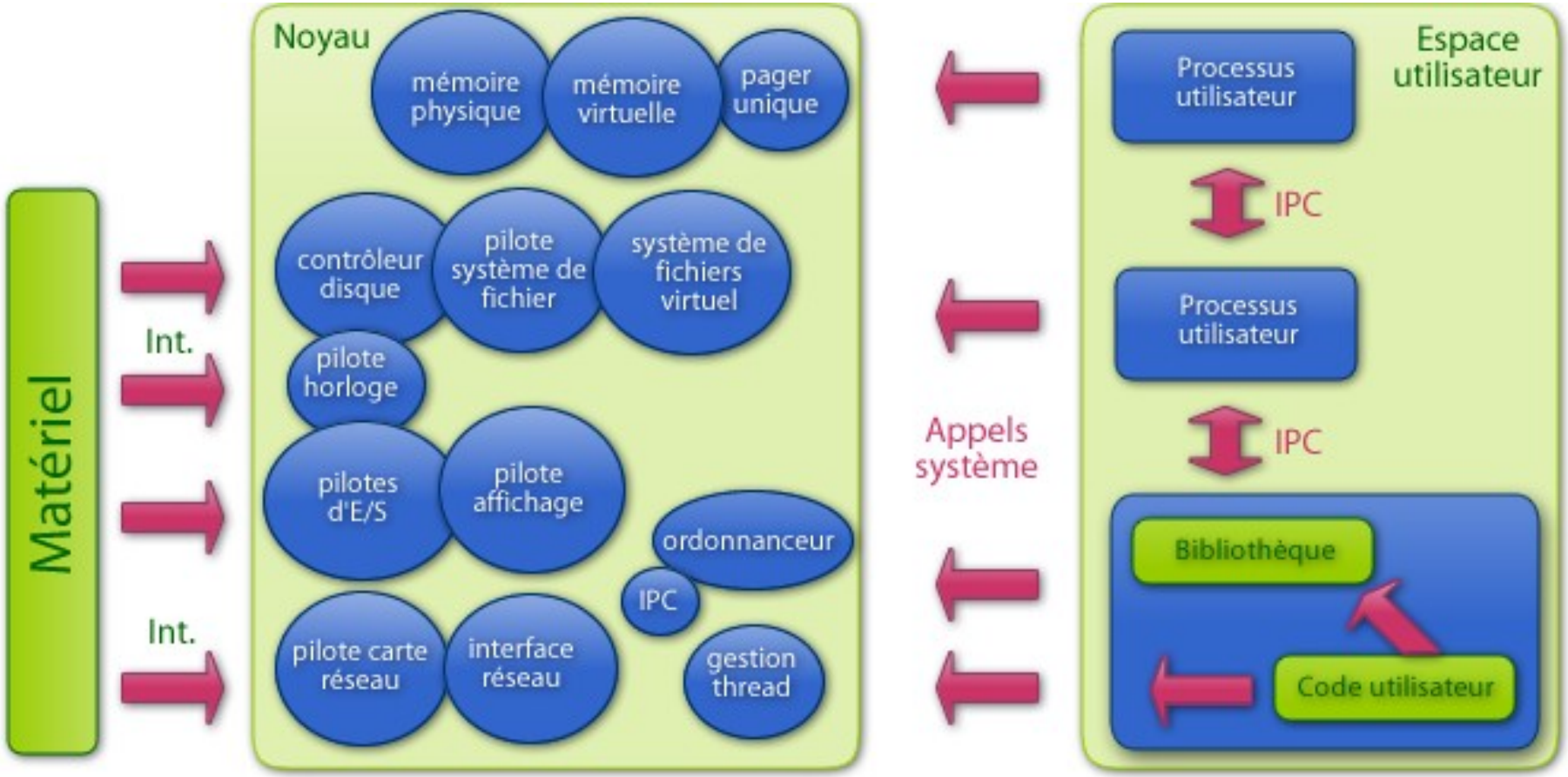
Matériel compatible

Matériel compatible

- Longtemps ce fut le problème majeur de Linux
- Évolution depuis l'adoption de Linux par des grands comptes
  - Disponibilité des pilotes (compatibilité matérielle)
  - Disponibilité des outils (exemple : outils FPGA)
  - Parfois (de + en + rare) pas de support technique « officiel » pour Linux
- Choix sur le matériel
  - Type de CPU : ARM, x86, PPC mais aussi Nios 2, MicroBlaze
  - MMU ou pas ?
  - Format de cartes : PC/104, SBC, modules

Composants logiciels d'une distribution

Architecture générale (rappel)



Architecture, suite

- Les éléments fondamentaux :
  - Le bootloader (indépendant du système)
  - Le noyau: en théorie, interface unique avec le matériel, API de programmation spécifique (modules Linux)
  - Le «root filesystem» (root-fs): les commandes et fichiers système, communs à (presque) toutes les versions d'UNIX (API standard => POSIX)
  - Un système Linux est obligatoirement l'association noyau + root-fs
  - Embarquer: optimiser le noyau + construire un root-fs léger

open wide INGENIERIE

Le bootloader

- Sur x86, le BIOS démarre un *bootloader* comme GRUB ou LILO
- Pour les autres architectures, le bootloader fait office de BIOS
  - U-Boot (le plus répandu)
  - Red Boot (Red Hat)
  - PPC Boot
  - CFE
- Le bootloader démarre le noyau Linux (partie statique)

Le chargement du noyau Linux

- Chargement de la partie statique du noyau:
  - `vmlinuz`
  - `zImage/bzImage`
  - `uImage` (U-Boot)
- Initialisation du CPU et services fondamentaux (ordonnanceur, gestion mémoire, disque, flash, ...)
- Montage du root-fs et démarrage des services et applications

open wide
INGENIERIE

Création d’une distribution embarquée

Différentes approches

- Utiliser une produit d'éditeur (Wind River, MV, ...) → € €€
- Adapter une distribution Linux classique

  - Limité au niveau matériel
  - Empreinte mémoire importante
  - Cas très particulier
- Créer la distribution « from scratch »

  - Complexe
  - Difficile/impossible à industrialiser:
    - gestion des dépendances
    - évolutions
- Utiliser un outil de génération : Buildroot, OpenEmbedded, OpenWrt, LTIB

open wide INGENIERIE

Les éléments nécessaires

- Chaîne de compilation croisée :
  - Gcc
  - Gas
  - Gld
  - LibC
  - ...
- Bootloader (U-Boot)
- Noyau Linux adapté
- Commandes Linux (`sh`, `ls`, `cp`, etc.)
- Outil de génération (BR)

Chaîne de compilation croisée

- Utiliser un compilateur binaire :
  - ELDK: http://www.denx.de/wiki/DULG/ELDK
  - Code Sourcery : https://sourcery.mentor.com/sgpp/lite/arm/portal/release1803
  - Installation simple
  - Support (payant) possible
  - Configuration connue => support par les forums
- Construire un compilateur
  - Crosstool-NG
  - Buildroot / OpenEmbedded

U-Boot, introduction

- Universal Bootloader
- Développement géré par DENX Software
  http://www.denx.de/wiki/DULG/Manual
- Support d'un grand nombre de cartes ARM, PowerPC, MIPS, SH4, ...
- Basé sur un principe de variables d'environnement (`setenv, printenv, saveenv`, ...)
- Possibilité/nécessité d'écrire des macros → chargées depuis le PC hôte
- Support d'adresses statiques ou DHCP, protocole TFTP
- Support des flash NOR, NAND
- Nouvelle version U-Boot v2 → Barebox

Busybox

- GNU/Linux basé sur « coreutils » est trop volumineux pour l'embarqué

```
$ ls -l /bin/bash
-rwxr-xr-x 1 root root 877480 21 mai    2010 /bin/bash
```

- Busybox remplace la majorité des commandes Linux par des versions « réduites »

```
$ ls -l /bin/busybox
-rwsr-xr-x 1 root root 670856 15 mars  09:45 /bin/busybox
```

- 95 % des OS « Linux embarqué » utilisent Busybox
- Simple, léger, portable
- Diffusé sous licence GPLv2

open wide
INGENIERIE

Outils

Outils de développement

- Intégration des outils GNU (compilation, mise au point) dans un EDI (Environnement de Développement Intégré)
  - Eclipse + plugin CDT (C/C++)
  - QtCreator
- Produits commerciaux liés à un éditeur (Eclipse + outils)
  - Workbench (Wind River)
  - DevRocket (Montavista)
- Libre ou commercial ?
  - Outils additionnels intégrés (LTT, KGDB, ...)
  - Support technique

Utilisation des outils de production

- 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 :-)
- Peut produire la chaîne croisée
- Produit les différents éléments de la distribution
  - Image du bootloader (`u-boot.bin`)
  - Noyau Linux (zImage, uImage)
  - Image du root-filesystem (`rootfs.jffs2, ...`)

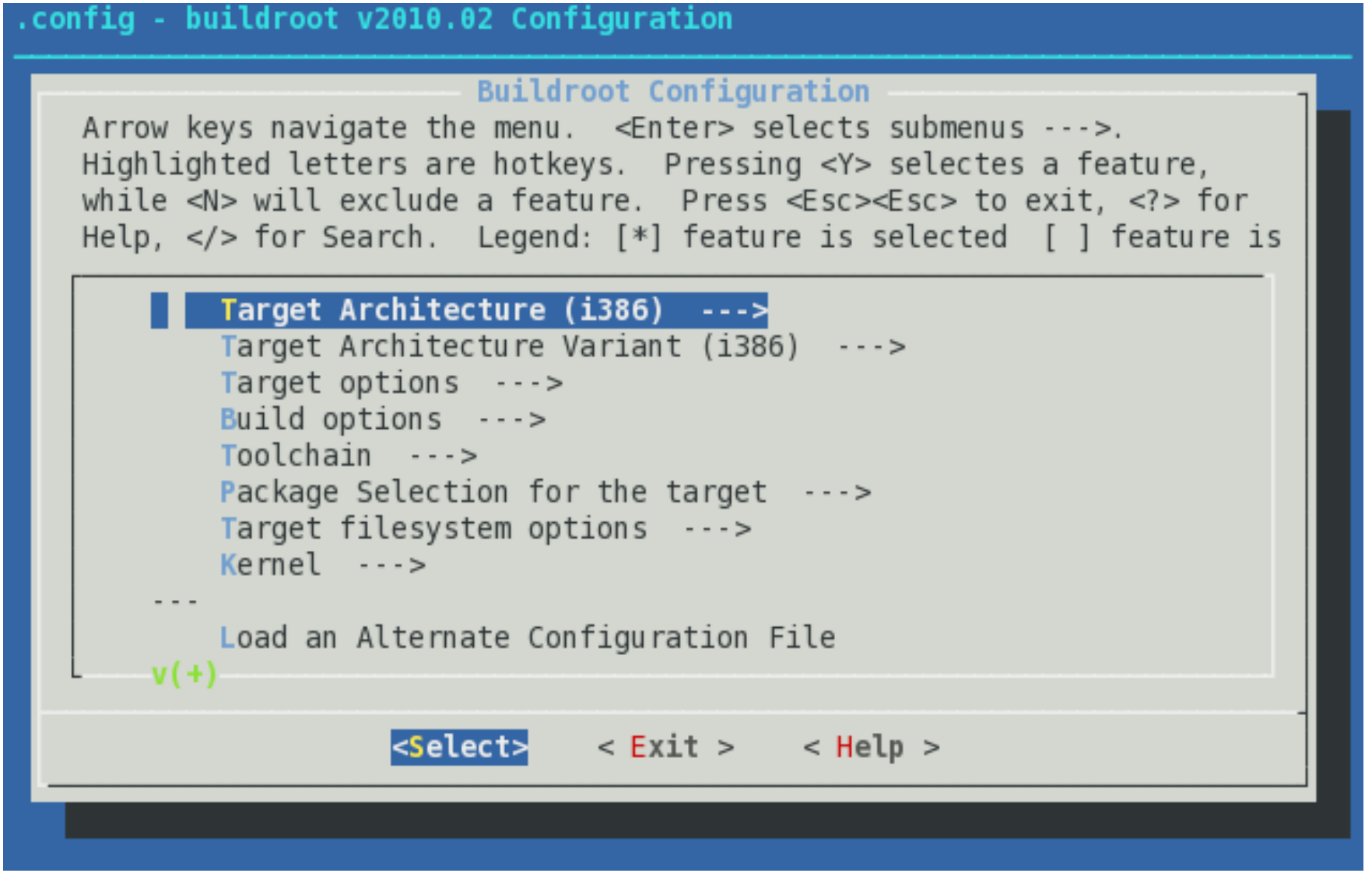
Quelques outils de production

- 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, PTXdist, ...

OpenEmbedded

- 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 ou l'on gère un grand nombre de configurations
- Gère la notion de paquet binaire, contrairement à BR

Buildroot


.config - buildroot v2010.02 Configuration
Buildroot Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> selectes a feature,
while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for
Help, </> for Search. Legend: [*] feature is selected [ ] feature is
Target Architecture (i386) --->
Target Architecture Variant (i386) --->
Target options --->
Build options --->
Toolchain --->
Package Selection for the target --->
Target filesystem options --->
Kernel --->

Load an Alternate Configuration File
v(+)
<Select>
< Exit >
< Help >

Buildroot, histoire

- Lié au projet uClibc (micro-C-libc) : libC plus légère 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
- Pas de version « officielle » avant 2009

open wide INGENIERIE

Buildroot aujourd'hui

- Repris en 2009 par Peter Korsgaard et Thomas Petazzoni
- Une version officielle tous les 3 mois: 2009.02, ..., 2011.02
- Projet géré sous Git
- Documentation largement améliorée
- Plus de 300 composants adaptés
- Support CPU x86, ARM, PowerPC, SH4, ...
- Il est assez « simple » d'ajouter un support de carte.

Configuration de Buildroot

- Type de CPU + variante, ex: arm puis arm920t
- Options de la cible
- Choix de la chaîne de compilation: interne (uClibc) ou externe (Glibc, Eglibc, ...), Crosstool-NG
- Composants de la distribution (répertoire package) => Qt, DirectFB, ...
- Noyau Linux et bootloader
- Formats des images du root-filesystem
  - JFFS2, CRAMFS, SQUASHFS → flash
  - EXT2, CPIO → ramdisk
  - TAR → NFS-Root