

Linux et les Logiciels Libres dans le domaine de l'embarqué

Thomas Petazzoni
Bootlin
<https://bootlin.com>



Intervenant

- ▶ Thomas Petazzoni
- ▶ Ingénieur Linux embarqué chez Bootlin
 - ▶ Spécialisé Linux embarqué : services de développement et formation
- ▶ Impliqué dans la communauté
 - ▶ Développeur Buildroot
 - ▶ Fondateur et membre CA de Toulibre
 - ▶ Membre CA de l'April
 - ▶ Créateur et animateur de l'Agenda du Libre
 - ▶ Co-auteur de MapOSMatic.org

Embarqué ?

- ▶ Recouvre des systèmes de types très différents
- ▶ Frontière floue avec les systèmes « classiques »
- ▶ Produits de grande consommation
 - ▶ Routeurs personnels, lecteurs de DVD, appareils photos numériques, GPS, caméscopes, téléphones, micro-onde, four
- ▶ Produits industriels
 - ▶ Commande de robot, alarmes, systèmes de surveillance, contrôle de machines, voiture, avion, satellite

Linux embarqué

- ▶ Le monde du Logiciel Libre offre toute une palette d'outils pour le développement de systèmes embarqués
- ▶ Émergence depuis ~10 ans dans l'embarqué
- ▶ Linux embarqué = Utilisation du noyau Linux et de composants Logiciels Libres au sens large pour faire fonctionner un système embarqué
- ▶ Convient pour les systèmes embarqués «complexes» qui ont besoin d'un OS
- ▶ Nombreux avantages liés à l'aspect libre

Linux Embarqué

- ▶ Parts de marchés actuelles des OS embarqués
 - ▶ OS propriétaire: 39%
 - ▶ Linux embarqué gratuit: 29%
 - ▶ Linux embarqué avec support commercial: 11%
 - ▶ OS maison: 7%
 - ▶ Pas d'OS: 11%
- ▶ Pour les projets futurs
 - ▶ Linux embarqué gratuit: 71%
 - ▶ Linux embarqué avec support commercial: 16%
 - ▶ OS propriétaire: 12%
 - ▶ OS maison: 1%
- ▶ Source: Venture Development Corp, octobre 2007

Avantages

- ▶ **Réutilisation de composants existants pour le système de base, permet de se focaliser sur sa valeur ajoutée**
 - ▶ On ne réinvente pas une pile réseau, une pile USB, des bibliothèques de base à chaque fois
 - ▶ Plateforme « Linux » normale, qui permet aussi de réutiliser les compétences
- ▶ Composants de bonne qualité
 - ▶ Mais il faut bien les choisir : solidité de la communauté, qualité du composant, etc.
- ▶ **Contrôle complet sur les composants, modifications sans contraintes**
 - ▶ Un aspect très important dans le domaine de l'embarqué, pour pouvoir intégrer autant que nécessaire, et corriger des problèmes en dépendant le moins possible de fournisseurs extérieurs

Avantages

- ▶ **Grande modularité et configurabilité des composants**
 - ▶ Permet de faire un système entièrement sur mesure et répondre à des besoins et contraintes variés
- ▶ Support de la communauté: tutoriels, listes
 - ▶ La communauté ne va pas développer le produit à votre place. Mais en contribuant à la communauté, celle-ci vous le rendra en aide et conseil
- ▶ Faible coût, et notamment pas de royalties par unité vendue
 - ▶ Peut-être capital quand le nombre d'unités produite est très grand
 - ▶ Ne pas négliger les coûts d'ingénierie, de formation, etc.
- ▶ Potentiellement moins de problèmes juridiques
 - ▶ Quelques licences couvrent 95% des logiciels libres
- ▶ Accès plus facile aux logiciels et outils

Produits grand public

- ▶ GPS
 - ▶ Tomtom et Garmin
- ▶ Routeurs personnels
 - ▶ Linksys, Freebox, Livebox
- ▶ PDA
 - ▶ Zaurus, Nokia N8x0
- ▶ Téléviseurs, caméscopes, lecteurs de DVDs
 - ▶ Sony, Phillipps
- ▶ Stockage réseau, disques durs multimédia
- ▶ Téléphone
 - ▶ Nokia N900, OpenMoko, Palm Pre
 - ▶ Android est du «faux» Linux embarqué

Embarqué ?



Domaine industriel

- ▶ Utilisation dans des domaines extrêmement variés
- ▶ Quelques exemples...

Éolien



Parc-mètre



Terminal de paiement



Déneigement



Ferroviaire



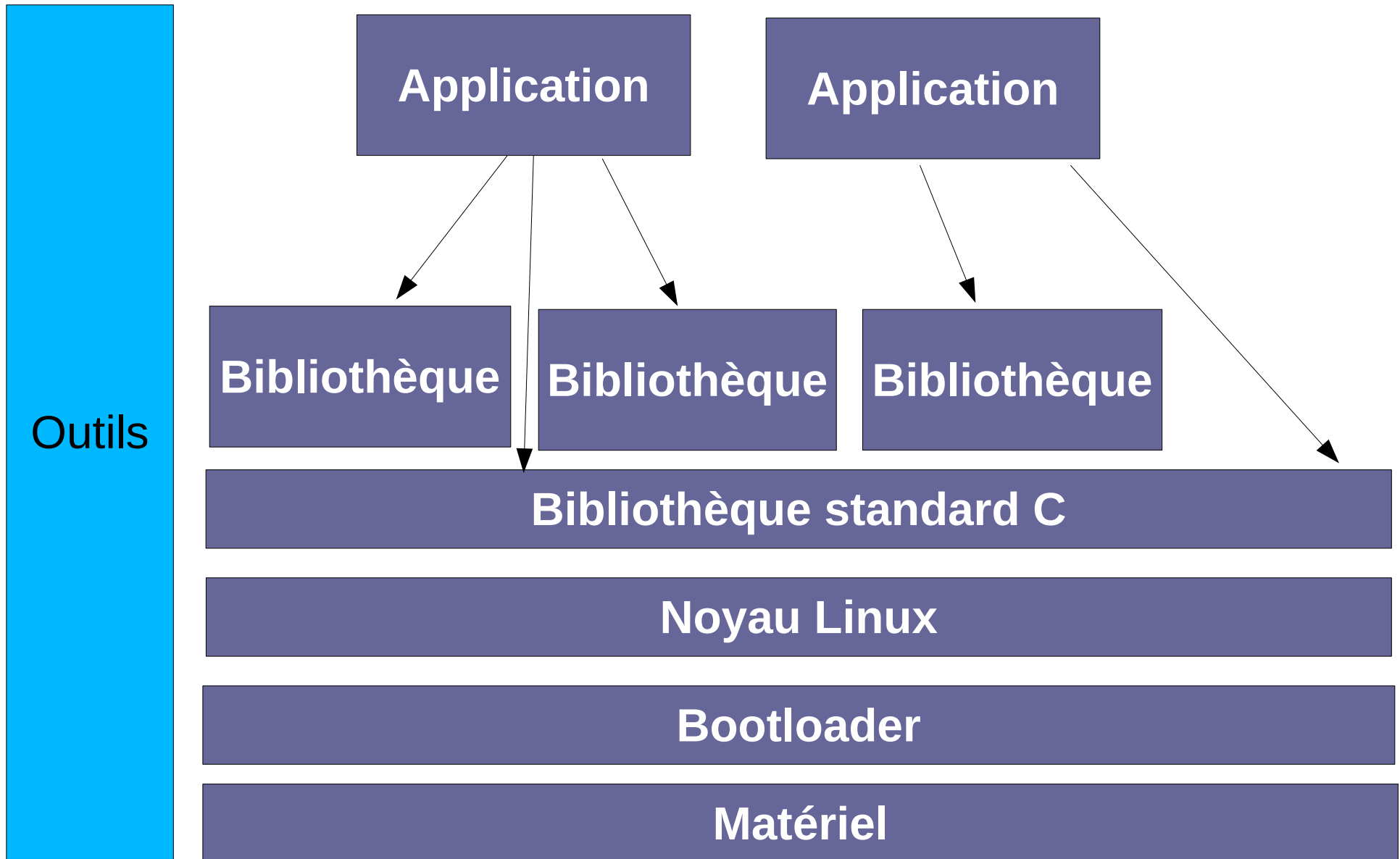
Avion



Derrière...



Architecture de base



Matériel pour l'embarqué

- ▶ Le matériel des systèmes embarqués est souvent différent de celui d'un système classique
 - ▶ Architecture processeur différente. Souvent ARM, MIPS ou PowerPC. x86 est aussi utilisé.
 - ▶ Stockage sur mémoire Flash, de type NOR ou NAND, de capacité souvent relativement réduite (quelques Mo à quelques centaines de Mo)
 - ▶ Capacité mémoire réduite (quelques Mo à quelques dizaines de Mo)
 - ▶ De nombreux bus d'interconnexion peu courants sur le desktop: I2C, SPI, SSP, CAN, etc.
- ▶ Cartes de développement à partir d'une centaine d'Euros
 - ▶ Souvent utilisées comme base pour le design final de la carte qui sera utilisée

Besoins minimaux

- ▶ Un processeur supporté par le compilateur gcc et le noyau Linux
 - ▶ Processeur 32 bit
 - ▶ Les processeurs sans MMU sont également supportés, au travers du projet uClinux
- ▶ Quelques mega-octets de RAM, à partir de 4 Mo, 8 Mo sont nécessaires pour pouvoir faire vraiment quelque chose.
- ▶ Quelques mega-octets de stockage, à partir de 2 Mo, 4 Mo pour faire vraiment quelque chose.
- ▶ Linux n'est donc pas fait pour les petits micro-contrôleurs qui ne possèdent que quelques dizaines ou centaines de Ko de Flash et de RAM
 - ▶ Sur le métal, pas d'OS
 - ▶ Systèmes réduits, type FreeRTOS, RTEMS, etc.

Chaîne de cross-compilation

- ▶ L'outil indispensable pour le développement embarqué sur des architectures non-x86.
- ▶ Contient des outils
 - ▶ S'exécutant sur une machine hôte (la machine du développeur, généralement x86)
 - ▶ Générant/manipulant du code pour machine cible (généralement non x86)
- ▶ « Outils binaires »
 - ▶ Binutils
 - ▶ Ld, as, nm, readelf, objdump, etc.
 - ▶ Bibliothèque standard C
 - ▶ glibc, uClibc ou eglibc
 - ▶ Compilateur C/C++
 - ▶ gcc
 - ▶ Bibliothèques mathématiques
 - ▶ gmp, mpfr
 - ▶ Débogueur
 - ▶ gdb

Bootloader

- ▶ Sur PC : LILO ou Grub
 - ▶ Le BIOS fait une bonne partie du travail et met à disposition du bootloader des routines pour le chargement de données depuis le disque
- ▶ Sur les architectures embarquées, pas de BIOS
 - ▶ Le bootloader doit tout faire, y compris l'initialisation du contrôleur mémoire
- ▶ Bootloader le plus courant: U-Boot
 - ▶ Très portable
 - ▶ Nombreux drivers
 - ▶ Nombreuses fonctionnalités (shell, scripting, etc.)

Noyau Linux

- ▶ Composant essentiel d'un système embarqué
- ▶ Les éléments de base du système: gestion des processus, de la mémoire, systèmes de fichiers, protocoles réseau, etc.
- ▶ Contient les pilotes pour la plupart des périphériques
- ▶ Le noyau distingue trois niveaux pour le support du matériel embarqué
 - ▶ L'architecture: ARM, MIPS, PowerPC
 - ▶ Le processeur: Samsung SC2442 par ex.
 - ▶ La machine: OpenMoko Freerunner

Noyau Linux

- ▶ Le noyau est le plus souvent porté sur une carte par le vendeur de celle-ci, sinon il faut s'adresser à une société spécialisée, ou mettre les mains dans le camboui.
- ▶ Un fichier de configuration est fourni pour chaque machine, il est personnalisable
- ▶ Après compilation, le noyau c'est
 - ▶ Une image binaire (le noyau lui-même), le plus souvent compressée, d'une taille de ~600 Ko à plusieurs Mo. C'est cette image qui est chargée et exécutée par le bootloader
 - ▶ Optionnellement des modules noyau
- ▶ Adaptations particulières à l'embarqué
 - ▶ Temps réel
 - ▶ Gestion de l'énergie
 - ▶ Stockage

Bibliothèque standard C

- ▶ La bibliothèque de base qui s'intercale entre d'un côté toutes les autres bibliothèques et applications et d'un autre côté le noyau
- ▶ Elle fait partie de la chaîne de cross-compilation
- ▶ Trois solutions
 - ▶ GNU Libc
 - ▶ uClibc
 - ▶ eglibc
- ▶ Avec la libc sur un système embarqué, on a déjà une API de programmation “riche” pour des applications non-graphiques
 - ▶ Threads, IPC, entrées-sorties, réseau, etc.

Busybox

- ▶ Besoin d'un ensemble d'outils de base pour la cible
- ▶ cp, ls, mv, mkdir, rm, tar, mknod, wget, grep, sed et tous les autres
- ▶ Une solution: utiliser les outils GNU classiques
 - ▶ fileutils, coreutils, tar, wget, etc.
 - ▶ Inconvénient: beaucoup d'outils, pas conçus pour l'embarqué
- ▶ Une meilleure solution: Busybox
 - ▶ Tous les outils dans un seul programme binaire
 - ▶ Des outils aux fonctionnalités réduites... et à taille réduite
 - ▶ Extrêmement configurable
 - ▶ Des liens symboliques pour les utiliser comme d'habitude
 - ▶ <http://www.busybox.net>
 - ▶ Utilisé dans de très nombreux de produits du marché

Bibliothèques et outils

- ▶ En théorie, toutes les bibliothèques et tous les outils libres peuvent être cross-compilés et utilisés sur une plateforme embarquée
 - ▶ Une fois le système en place, c'est juste du Linux !
 - ▶ Des milliers de composants à sélectionner et à réutiliser : bibliothèques graphiques, bibliothèques et outils réseau, composants système, bibliothèques multimédia, langages, etc.
- ▶ En pratique, la compilation croisée n'est pas toujours aisée, car pas prévue par les développeurs originaux
 - ▶ Difficulté d'intégration
- ▶ Des outils plus spécifiquement destinés aux plateformes limitées

Outils de construction

- ▶ Plusieurs approches pour la construction d'un système embarqué
 - ▶ À la main
 - ▶ Pénible, peu reproductible, difficulté de trouver les bonnes options, d'appliquer les bons patches, etc.
 - ▶ Par des outils de construction
 - ▶ Buildroot
 - ▶ OpenEmbedded
 - ▶ PTXdist
 - ▶ Par des distributions
 - ▶ Gentoo Embedded
 - ▶ Debian Embedded

Tâches

- ▶ Réalisation du Board Support Package (BSP)
 - ▶ Adaptation du chargeur de démarrage et du noyau à la plateforme matérielle. Permet ensuite de faire fonctionner n'importe quel système Linux
- ▶ Intégration du système
 - ▶ Selon l'application ciblée par le système, intégration d'un certain nombre de composants: bibliothèques, applications, etc.
- ▶ Développement de l'application
 - ▶ Un développement applicatif Linux traditionnel, en reposant sur les briques sélectionnées

Applications industrielles

- ▶ Dans de nombreuses applications industrielles, le système est « seulement » chargé de contrôler un équipement et de communiquer avec l'extérieur
- ▶ Un tel système comporte en général relativement peu de composants :
 - ▶ Noyau
 - ▶ BusyBox
 - ▶ Bibliothèque standard
 - ▶ Applications reposant directement sur la bibliothèque standard C, parfois utilisant les possibilités temps-réel du noyau
 - ▶ Parfois un serveur Web pour le contrôle à distance, ou un autre serveur implémentant un protocole spécifique

Cadre photo numérique

- ▶ Exemple pris d'une conférence de Matt Porter, Embedded Alley à ELC 2008
- ▶ Matériel: SoC ARM avec DSP, audio, LCD 800x600, MMC/SD, NAND, boutons, haut-parleurs
- ▶ Le cadre photo doit pouvoir
 - ▶ Afficher des photos à l'écran
 - ▶ Détecter l'insertion de cartes SD et notifier les applications, de manière à ce qu'elles puissent afficher les photos présentes sur la carte
 - ▶ Interface 3D moderne avec transitions
 - ▶ Navigation avec les boutons
 - ▶ Lecture audio (MP3, tags ID3, playlist)
 - ▶ Redimensionnement et rotation des photos

Cadre photo : composants (1)

- ▶ Système de base
 - ▶ Composants présents dans l'intégralité des systèmes Linux embarqué
 - ▶ Chargeur de démarrage U-Boot
 - ▶ Noyau Linux, avec notamment les pilotes pour SD/MMC, framebuffer, son, boutons
 - ▶ Busybox
 - ▶ Système de construction: OpenEmbedded

Cadre photo : composants (2)

- ▶ Gestion des évènements pour l'insertion de carte SD
 - ▶ Udev, qui reçoit des évènements du noyau, crée des fichiers périphériques et envoie des évènements à HAL
 - ▶ HAL, qui maintient une base de données des périphériques disponibles et offre une API via D-Bus
 - ▶ D-Bus pour connecter HAL avec l'application. L'application s'inscrit aux évènements HAL via D-Bus et est notifié lorsqu'ils surviennent

Cadre photo : composants (3)

- ▶ Affiche d'images
 - ▶ libjpeg pour décoder les images
 - ▶ jpegtran pour le redimensionnement et la rotation
 - ▶ FIM (Fbi Improved) pour le *dithering*
- ▶ Support MP3
 - ▶ libmad pour la lecture
 - ▶ libid3 pour la lecture des tags ID3
 - ▶ libm3u pour le support des listes de lecture
 - ▶ Utilisation de composants fournis par le vendeur du CPU pour utiliser le DSP

Cadre photo : composants (4)

- ▶ Interface 3D
 - ▶ Vincent, une implémentation libre d'OpenGL ES
 - ▶ Clutter, une bibliothèque fournissant une API de haut-niveau pour l'implémentation d'applications 3D
- ▶ L'application elle-même
 - ▶ Gère les évènements liés à l'insertion/retrait de média
 - ▶ Utilise les bibliothèques JPEG pour décoder et effectuer le rendu des images
 - ▶ Reçoit des évènements des boutons
 - ▶ Affiche une interface 3D basée sur OpenGL/Clutter
 - ▶ Gère une configuration définie par l'utilisateur
 - ▶ Joue la musique avec les bibliothèques MP3
 - ▶ Affiche un diaporama des photos

En savoir plus

- ▶ Supports de formation de Bootlin, disponibles sous licence libre
 - ▶ Intégration de système Linux embarqué
 - ▶ Développement de code noyau
- ▶ Livre « Building Embedded Linux Systems », O'Reilly, 2008
- ▶ Livre « Embedded Linux Primer », Prentice Hall, 2006
- ▶ Embedded Linux Wiki, <http://elinux.org>
- ▶ Conférences ELC et ELCE, vidéos mises à disposition par Bootlin
- ▶ Assez peu de littérature en français
 - ▶ Articles dans Linux Magazine
 - ▶ Livre « Linux Embarqué » par Pierre Fichoux, un peu obsolète car publié en 2002