Des Logiciels Libres à Linux embarqué

Thomas Petazzoni
thomas.petazzoni@free-electrons.com
Mardi 1er décembre 2009

Intervenant

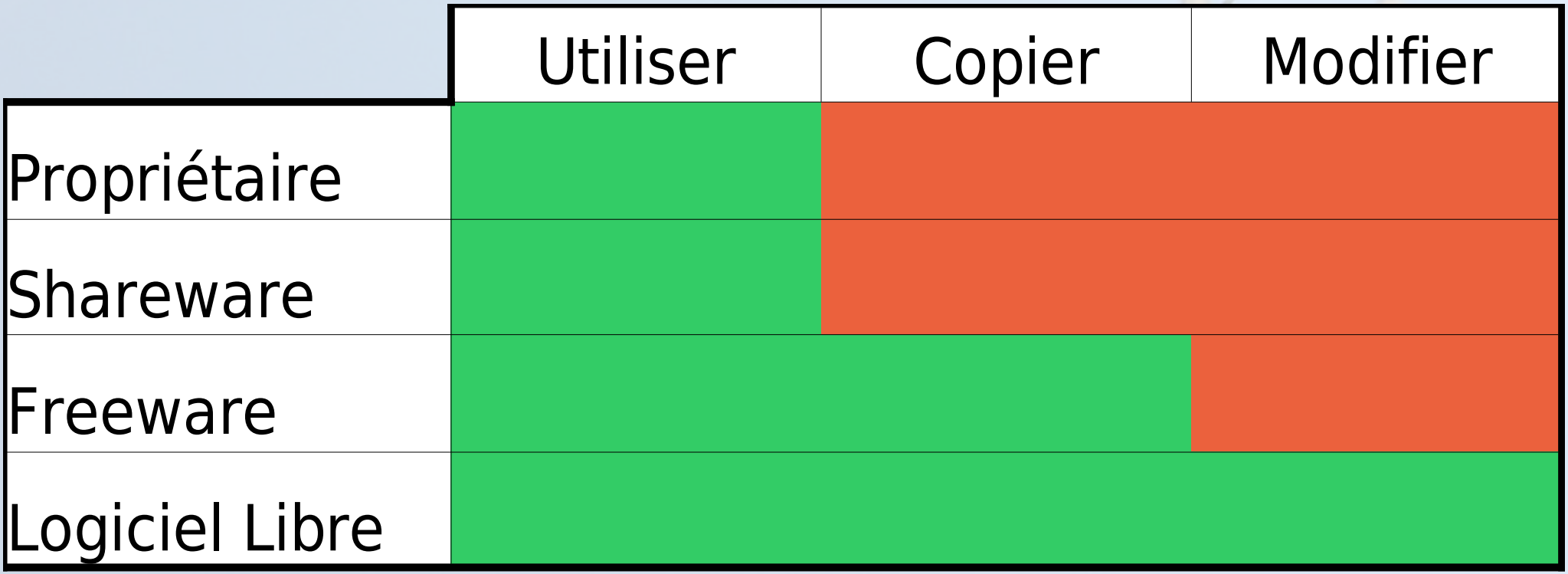
- Thomas Petazzoni
  - Ingénieur en informatique, formateur et développeur spécialisé Linux embarqué
  - dans la société Free Electrons
  - Utilisateur, développeur et promoteur des Logiciels Libres depuis 2000
  - Impliqué dans la communauté
    - Créateur et animateur de l'Agenda du Libre
    - Membre du Conseil d'Administration de l'April
    - Fondateur et animateur de Toulibre, association de promotion du Libre à Toulouse
    - Contributeur dans quelques projets libres

- **Logiciel Libre ?**
- Parts de marché et intérêts dans l'embarqué
- Le monde Linux embarqué

Principes du Logiciel Libre

- Quatre libertés
  - liberté d'**utilisation** d'un programme
  - liberté d'**étudier** le fonctionnement d'un programme
  - liberté de **modifier** un programme
  - liberté de **distribuer** un programme
- Penser à la libre expression et à la liberté, pas à la gratuité
- Disponibilité du *code source*
- S'oppose à « logiciel propriétaire »

Libre, propriétaire et autres



Droit d'auteur et logiciel

- Comme toutes les créations, les programmes d'ordinateur sont automatiquement protégés par le **droit d'auteur**
- Accorde un monopole, choix de l'exploitation
- Le droit d'auteur garantit qu'on ne peut:
  - copier un programme pour le donner ou le vendre, (essayer de) le modifier, l'utiliser en dehors des clauses prévues par sa licence
- Le droit d'auteur n'interdit en revanche pas d'écrire un nouveau programme:
  - Aux fonctionnalités similaires, compatible au niveau des formats de communication et de données, etc.

Les licences

- En se basant sur le droit d'auteur, les licences d'utilisation déterminent les droits et devoirs des utilisateurs:
  - « Licence propriétaire » : une liberté d'utilisation, limitée
  - « Licence libre » : organisation de la diffusion du programme
- Les Logiciels Libres ne sont pas «libres de droit» ou dans le «domaine public»

Copyleft

- La liberté des uns ne doit pas restreindre la liberté des autres : ce qui est libre doit le rester
- Les licences de logiciels libres garantissent les quatre libertés et pour certaines garantissent la **persistance** des quatre libertés
- Notion de ***copyleft***
- GPL: « *Création d'un pot commun auquel chacun peut ajouter mais rien retirer* » - Eben Moglen

Licence GPL

- *GNU General Public License*, **GPL**, principale licence de la FSF
- Utilisée par 60-70% des Logiciels Libres
- Offre les quatre libertés fondamentales du Logiciel Libre
- Met en oeuvre le principe du *copyleft* : les œuvres dérivées doivent être distribuées sous la même licence
- GPL version 3 parue en juin 2007

Licence GPL

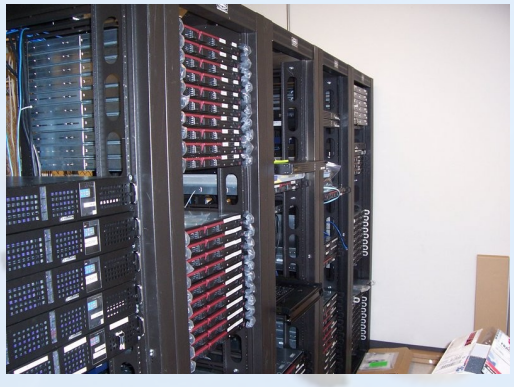
- Impose la distribution du code source seulement à ceux à qui on distribue le code binaire
- N'impose pas la distribution du code source s'il n'y a pas distribution du code binaire:
  - utilisation en interne, service Web
- N'impose pas la distribution du code source de toute l'infrastructure logicielle : seulement des œuvres dérivées
- Autorise la commercialisation des logiciels

Autres licences

- Licence LGPL et GPL avec exception Classpath
  - utilisées pour les bibliothèques, permet leur utilisation dans un logiciel propriétaire
- Licences BSD
  - offre les 4 libertés
  - pas de copyleft
- Licences X11, Cecill, Cecill-B, Cecill-C, Perl, Ruby, Apache...
- Nombreuses licences, mais moins nombreuses que dans le monde propriétaire

Serveurs

- Domaine de prédilection de GNU/Linux
- Web: Apache
  - LAMP: Linux, Apache, MySQL, PHP
- DNS: Bind
- Base de données: MySQL, PostgreSQL
- Courrier électronique: Sendmail, Postfix, Exim
- Fichiers et impressions: NFS, Samba
- Application: Jboss, Jonas, Tomcat, Zope
- Central téléphonique: Asterisk
- Supercalculateurs: 75% du TOP500



Poste de travail

- **Bureautique:** OpenOffice.org, Abiword, Gnumeric, Koffice...
- **Navigateur Web:** Firefox, Konqueror, Galeon
- **Courrier électronique:** Thunderbird, Evolution, Sylpheed Claws, Kmail...
- **Graphisme, PAO:** Gimp, Blender, Scribus, Inkscape, Sketch, NVU...
- **Comptabilité:** Grisbi, Gnucash
- **Multimédia**: Amarok, Totem, Rhythmbox, Mplayer, Videolan, Kino, Cinelerra
- **Environnements de bureau:** Gnome, KDE, XFCE...

Système GNU/Linux

- **Assemblage** de logiciels d'origines diverses
  - Noyau Linux
  - Projet GNU
  - Système graphique (X.org, KDE, Gnome, ...)
  - Applications (Firefox, OpenOffice, Gimp, ...)
  - Serveurs, outils
- Installé sous forme de **distribution**
  - Intégration des logiciels entre eux
  - Système de paquetage
  - Procédure d'installation

Web

- **Moteurs de Wiki**: MediaWiki, Wikini, DokuWiki, Xwiki, Twiki, MoinMoin, etc.
- **Moteurs de blogs, CMS**: Dotclear, Wordpress, SPIP, Joomla, Drupal, Lutece, Plone, Typo3, etc.
- **Webmails**: Horde IMP, OpenWebMail, RoundCube, Squirrelmail,etc.
- **Groupware**: phpGroupWare, Open-XChange, OpenGroupWare, eGroupWare, Hula, etc.

Métier

- Un domaine en expansion
  - ERPs: Compiere, TinyERP, Adampiere, ERP5, Ofbiz Neogia, etc.
  - OpenCascade: modélisation 3D et simulation numérique
  - CodeAster: analyse des structures et thermo-mécanique
  - SugarCRM, vTiger, OpenSourceCRM: gestion de contacts
  - Interchange, OsCommerce: commerce électronique
  - ...

85 % des entreprises ont adopté des Logiciels Libres

100% des entreprises en auront adopté d'ici un an

Étude Gartner novembre 2008

- Logiciel Libre ?
- **Parts de marché et intérêts dans l'embarqué**
- Le monde Linux embarqué

Embarqué ?

Un système embarqué peut être défini comme un système électronique et informatique autonome, qui est dédié à une tâche bien précise. Ses ressources disponibles sont généralement limitées. Cette limitation est généralement d'ordre spatial (taille limitée) et énergétique (consommation restreinte).

Wikipédia,
http://fr.wikipedia.org/wiki/Système_embarqué

Embarqué ?

- Une définition assez générale
  - 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, camescopes, téléphones, micro-onde, four
- Produits industriels
  - Commande de robot, alarmes, systèmes de surveillance, contrôle de machines, voiture, avion, satellite

Linux

- Lancé en 1991
- Un étudiant finlandais, Linus Torvalds
- Objectif: avoir un système de type Unix sur son ordinateur personnel Intel 386
- Devient portable en 1995
- En association avec des composants GNU et autres, constitue la base d'un système d'exploitation complet

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 Logiciel Libre de Linux et des composants

Avantages de Linux embarqué

- Réutilisation de composants existants pour le système de base, permet de se focaliser sur sa valeur ajoutée
  - Support matériel, support réseau, architecture de base du système
  - Plateforme Linux « normale »
- Composants de bonne qualité, pour certains
  - Utilisé sur des millions de machines
- Contrôle complet sur les composants, modifications sans contraintes
  - Code source disponible et modifiable

Avantages de Linux embarqué

- Modularité et configurabilité extrême
- Support de la communauté: tutoriels, listes
- Faible coût, et notamment pas de royalties par unité vendue
- Potentiellement moins de problèmes juridiques
  - Peu de licences libres
- Accès plus facile aux logiciels et outils
  - Pas de licence à acquérir pour pouvoir évaluer une solution

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

Linux Embarqué

- GPS: Tomtom et Garmin
- Routeurs personnels: Linksys
- PDA: Zaurus, Nokia N8x0
- Téléviseurs, camescopes, lecteurs de DVDs, etc: Sony
- Téléphone: Motorola, OpenMoko
- Train, avion, parc-mètres, éoliennes, paiement par C.B, etc.
- Et plein d'autres usages que l'on imagine même pas...

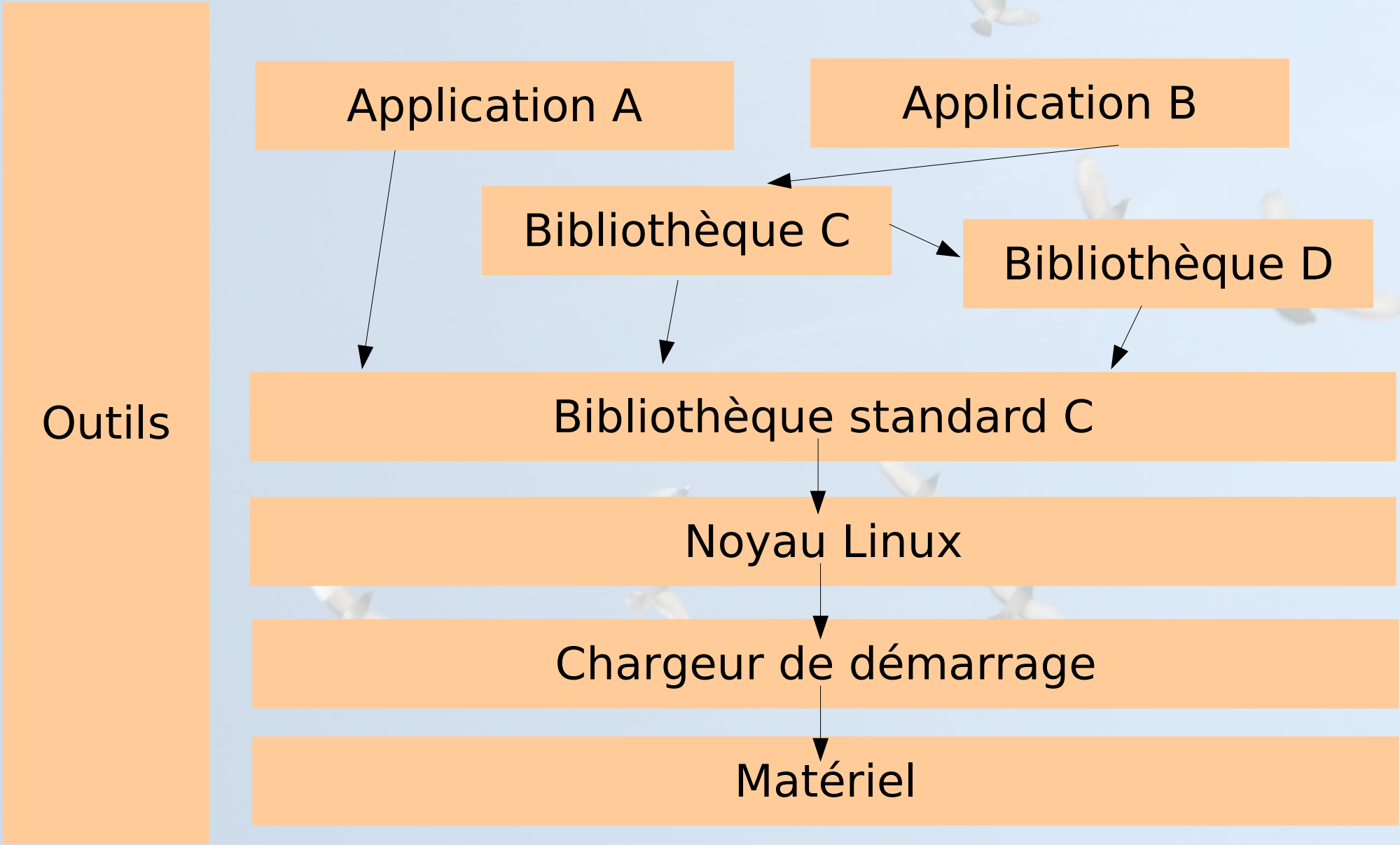
Embarqué ?

Ça fonctionne avec Linux, mais à quoi ça sert ?

À traire les vaches !

- Logiciel Libre ?
- Parts de marché et intérêts dans l'embarqué
- **Le monde Linux embarqué**

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.

Processeur

- Intégration beaucoup plus forte que dans le monde x86
  - Principe du System-on-Chip: intégration du cœur du processeur avec les périphériques sur une seule puce
- Des concepteurs de cœur
  - ARM, MIPS, PowerPC
- Des intégrateurs et fondeurs
  - Texas Instruments, Freescale, Atmel, Samsung, Cavium Networks, NEC, Toshiba, PMC-Sierra, Altera, Cisco, STMicroelectronics, etc.

Processeur

- Chaque processeur est un assemblage particulier d'un cœur et de périphériques
  - Grand choix, compris coût, performances, fonctionnalités
- Cartes de développement à partir d'une centaine d'Euros
  - Souvent utilisées comme base pour le design final de la carte qui sera utilisée

Exemples

- Picotux 100
  - ARM7 55 Mhz, Netsilicon NS7520
  - 2 Mo de Flash
  - 8 Mo de RAM
  - Ethernet
  - 5 GPIO
  - Série

- OpenMoko
  - ARM 920T 400 Mhz, Samsung 2442B
  - 2 Mo de Flash NOR
  - 128 Mo de RAM
  - 256 Mo de Flash NAND
  - Touchscreen 640x480, Bluetooth, GSM, série, GPS, son, deux boutons, Wifi, USB, etc.

Exemples

- Carte Calao Systems USB A9263
  - Processeur Atmel AT91SAM9263 à 180 Mhz
  - 64 Mo de RAM
  - 256 Mo de NAND
  - Ethernet
  - Série/JTAG/alim sur USB
  - USB host et device
  - Plein de cartes d'extension
  - ~ 150 €
- Carte Armadeus APF27
  - Processeur Freescale iMX27
  - 128 Mo de RAM
  - 256 Mo de NAND
  - Ethernet
  - Série
  - USB host
  - FPGA !

Émulation

- Qemu permet l'émulation de nombreuses architectures CPU
  - X86, bien sûr, mais aussi PowerPC, ARM, MIPS, SPARC, etc.
- Émulation complète: processeur, mémoire et périphériques
- Pour chaque architecture, plusieurs plateformes sont proposées
  - Pour ARM: Integrator, Versatile, PDA Sharp, Nokia N8x0, Gumstix, etc.
- Idéal pour commencer à bricoler autour de Linux embarqué, même si le contact avec le matériel est plus « fun »
- Web: http://bellard.org/qemu/

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
- Mémoire
  - 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 / RAM
  - Sur le métal, pas d'OS ou 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
- 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
  - Lors de la mise sous tension, le CPU commence l'exécution à une adresse fixée
  - Le design matériel fait en sorte qu'une partie de la Flash soit mappée à cette adresse
  - Le point d'entrée du bootloader est stocké à cette adresse dans la Flash: il prend le contrôle du matériel dès sa mise sous tension
  - Parfois un bootloader de premier niveau de petite taille

Bootloader: U-Boot

- Il existe de nombreux bootloaders pour les architectures non-x86, certains plus ou moins spécifiques à certaines architectures ou plateformes
- Le bootloader libre le plus utilisé est Das U-Boot
- Il supporte un grand nombre d'architectures, est aisément configurable et modifiable
- Fonctionnalités de base
  - Téléchargement du noyau et du système de fichiers par le réseau (protocole TFTP)
  - Protection, effacement et écriture sur la Flash
  - Exécution du noyau
  - Outils de diagnostics: lecture/écriture mémoire, test de périphériques, etc.
- http://www.denx.de/wiki/U-Boot

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, gestion de l'énergie, 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

Bibliothèque standard C

- La bibliothèque de base qui s'intercale entre d'un coté toutes les autres bibliothèques et applications et d'un autre coté le noyau
  - Elle fait partie de la chaîne de cross-compilation
- Trois solutions
  - GNU Libc, la version standard utilisée sur tous les systèmes desktop/serveur. Fonctionnalités complètes, mais grosse.
  - uClibc, une ré-écriture complète d'une libc plus simple, optimisée pour la taille et configurable en fonctionnalités
  - eglibc, une reprise de la GNU Libc ajoutant plus de flexibilité au niveau de la configuration
- 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

Busybox

- 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
- Des outils plus spécifiquement destinés aux plateformes limitées

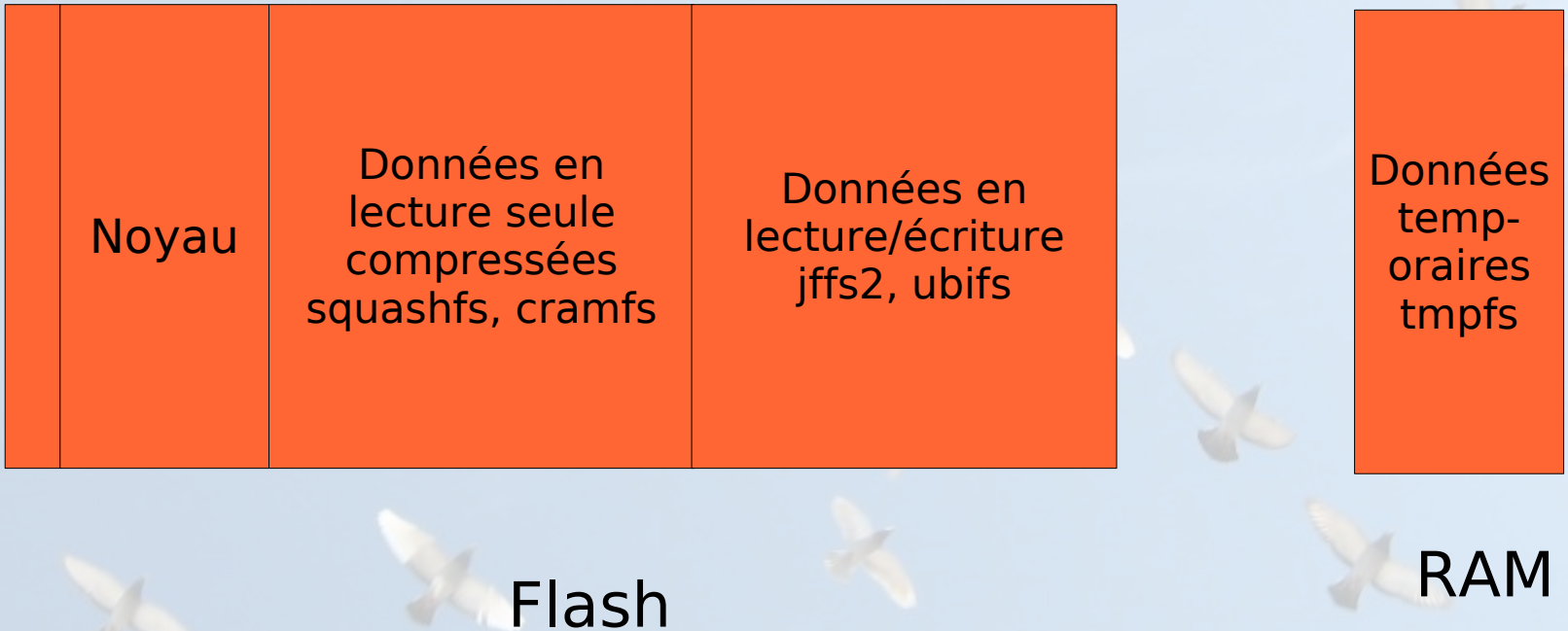
Outils de construction

- 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

Stockage

- Les Flash des systèmes embarqués sont généralement des flash brutes
- Aucun matériel ne s'occupe du « wear leveling », c'est à dire la répartition des écritures
- Des systèmes de fichiers spécifiques doivent être utilisés
- Aujourd'hui: JFFS2, UBIFS
- Autres systèmes de fichiers pour l'embarqué: squashfs et cramfs, pour les parties en lecture seule, tmpfs pour les données temporaires

Stockage


Noyau
Données en lecture seule compressées squashfs, cramfs
Données en lecture/écriture jffs2, ubifs
Données temp-oraires tmpfs
Flash
RAM

Temps réel

- Une contrainte dans certains systèmes embarqués est la prédictabilité du temps de réponse
  - Capture de données ou réaction suite à un évènement qui doivent avoir lieu dans un temps borné
- Nécessite un système dit « temps réel »
- Le temps réel n'a rien à voir avec la performance
  - Les bornes temporelles à garantir peuvent être larges, mais doivent être garanties
  - Un système temps réel est parfois globalement plus lent qu'un système à temps partagé classique

Temps réel et Linux

- Linux n'est pas conçu à l'origine comme un système temps réel, mais comme un système à temps partagé
  - Objectif: maximiser l'utilisation des ressources pour maximiser le rendement global des applications
- Deux approches pour apporter le temps réel au noyau
  - Linux-rt, une approche dans le noyau. Objectif: réduire au fur et à mesure les zones pendant lesquelles le noyau ne réagit pas à un évènement extérieur.
  - RTAI, Xenomai, une approche de co-noyau, où un mini-noyau temps réel se charge de traiter les évènements importants, en garantissant des temps de réponse. Linux tourne en tâche de fond.

Tâches

- Réalisation du Board Support Package
  - Adaptation du chargeur de démarrage et du noyau aux spécifités de la plateforme (drivers, etc.)
- Intégration du système
  - Sélection des composants appropriés et intégration dans un système Linux embarqué complet
- Développement des applications
  - Reposant sur les composants sélectionnés et offrant les fonctionnalités demandées

Exemples d'utilisation

- Deux exemples d'utilisation, avec les composants ré-utilisés
  - Applications industrielles
  - Cadre photo numérique

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 unn 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, créé 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 Free-Electrons, disponible sous licence libre
  - Intégration de système Linux embarqué
  - Développement de code noyau
- Livres
  - « Building Embedded Linux Systems », O'Reilly, 2008
  - « Embedded Linux Primer », Prentice Hall, 2006
- Embedded Linux Wiki, http://elinux.org
- Conférences ELC et ELCE, vidéos mises à disposition par Free-Electrons
- Assez peu de littérature en français
  - Articles dans Linux Magazine ou livre « Linux Embarqué » par Pierre Ficheux, un peu obsolète car publié en 2002

Questions ?