Informatique Embarquée
M2 / 2017

Distributions Linux
(pour systèmes embarqués)

Distribution Vs Système

- Système:
  - Système d'exploitation, Noyau
  - Linux: http://kernel.org/
  - Arbre des sources des différentes versions du noyau
- Distribution:
  - Ensemble des logiciels nécessaires pour une fonctionnalité complète et du noyau
  - Inclus: outils d'administrations et applications
    - Mount, fsck, ifconfig, init,...
    - Éditeurs, compilateurs,..

Versions du Noyau Linux

- https://en.wikipedia.org/wiki/Linux_kernel#Maintenance
- Versions: A.B.C[.D]
  - 2.4
  - 2.6
  - 3.x à partir de 07/ 2011
  - 4.x à partir de 04/2015
- Attention les distributions ajoutent des suffixes et fournissent leur propre noyau dérivé de l'arbre d'origine.
  - Ex: Linux debian 2.6.18-4-486

Distributions Linux « Classiques »

- Redhat
  - Fedora: gratuit, http://fedoraproject.org
  - RHEL: commercial, avec support
  - CentOS: « équivalent RHEL » gratuit, support communautaire, http://www.centos.org
- Suse
  - Suse: commercial, avec support
  - OpenSuse: gratuit http://www.opensuse.org/

Distributions Linux Classiques

- Debian
- Ubuntu
- Et beaucoup plus
  - http://en.wikipedia.org/wiki/Comparison_of_Linux_distributions
  - http://fr.wikipedia.org/wiki/Distribution_Linux

LSB: Linux Standard Base

- Résoudre les problèmes d'interopérabilité et de portabilité entre systèmes Linux
  - un ensemble de bibliothèques standards,
  - un nombre de commandes et d'utilitaires qui étendent le standard POSIX,
  - la structure de la hiérarchie du système de fichiers,
  - les différents run levels, *(historique)*
  - Le « packaging »: rpm (et pas deb)
  - et plusieurs extensions à X Window System.
- http://www.linuxfoundation.org/en/LSB

LSB

- Complète la norme SUS (Single Unix Specif.)
  - Single Unix Specification v4 / Posix
    - http://en.wikipedia.org/wiki/Single_UNIX_Specification
    - http://www.unix.org/what_is_unix/single_unix_specification.html
  - Sus définit une API
    - Application Programming Interface
- S'appuie sur des spécifications d'ABI
  - Spécifiques aux processeurs

API: Application Programming Interface

- Définit le prototype des fonctions
  - Ex: int open (const char * name, int oflags,...)
- Définit les fichiers d'inclusions (header files)
- Définit le comportement de la fonction
- Permet la portabilité source des applications après recompilation
- Ne permet pas la portabilité binaire des applications => ABI.
- Posix (sous-ensemble de SUS v3) définit une API

ABI: Application Binary Interface

- Permet la portabilité binaire des applications
  - Et donc dépendant du processeur ABI x86, ABI Sparc,...
- Format de fichiers binaires exécutables
  - ELF, format de debug, description des symboles, librairies,...
- Conventions invocations de fonctions
  - Passage de paramètres (registre / pile)
  - Retours de fonction

ABI: Application Binary Interface

- Éventuellement, invocation d'appels systèmes
  - Numéro associé, Passage paramètres entrée / sortie
  - Handler de signaux, etc...

Besoins pour systèmes embarqués

- Processeur pas forcément x86
- Contraintes mémoire
  - Pas forcément de support de MMU sur le processeur
  - Minimiser le nombre / taille des processus résidents en mémoire
  - Minimiser la taille du système de fichiers
    - Pas besoin d'installer emacs, gcc, firefox, ...
    - Besoins d'administration système réduits,
- Contraintes temporelles
  - Ordonnanceur et préemptabilité appropriés.

Distributions Commerciales pour Linux Embarqué

- https://fr.wikipedia.org/wiki/Linux_embarqu%C3%A9
- MontaVista Linux
  - Professional Edition
  - Carrier-Grade Edition (Carrier Grade Linux)
  - MobiLinux
- Wind-River:
  - Platform for General Purpose -Linux Edition
  - Platform for Consumer Device -Linux Edition
  - Platform for Network Equipments -Linux Edition (Carrier Grade Linux)

Distributions Commerciales pour Linux Embarqué

- TimeSys
- ELinos
- Et beaucoup plus encore
-
- Voir http://distrowatch.com/dwres.php?resource=links#embed

Distributions Open-Source pour Linux Embarqué

- Embedded Debian Project (arrêt fin 2014)
- Damn Small Linux
  - Basé sur Debian, File system < 50 Mb,
  - RAM: 16 à 24 Mb
- µCLinux (you-see-Linux)
  - Noyau pour processeurs sans MMU
- Yocto : https://www.yoctoproject.org/
- Mlinux, PeeWeeLinux, OpenWrt, ...
- Et beaucoup d'autres

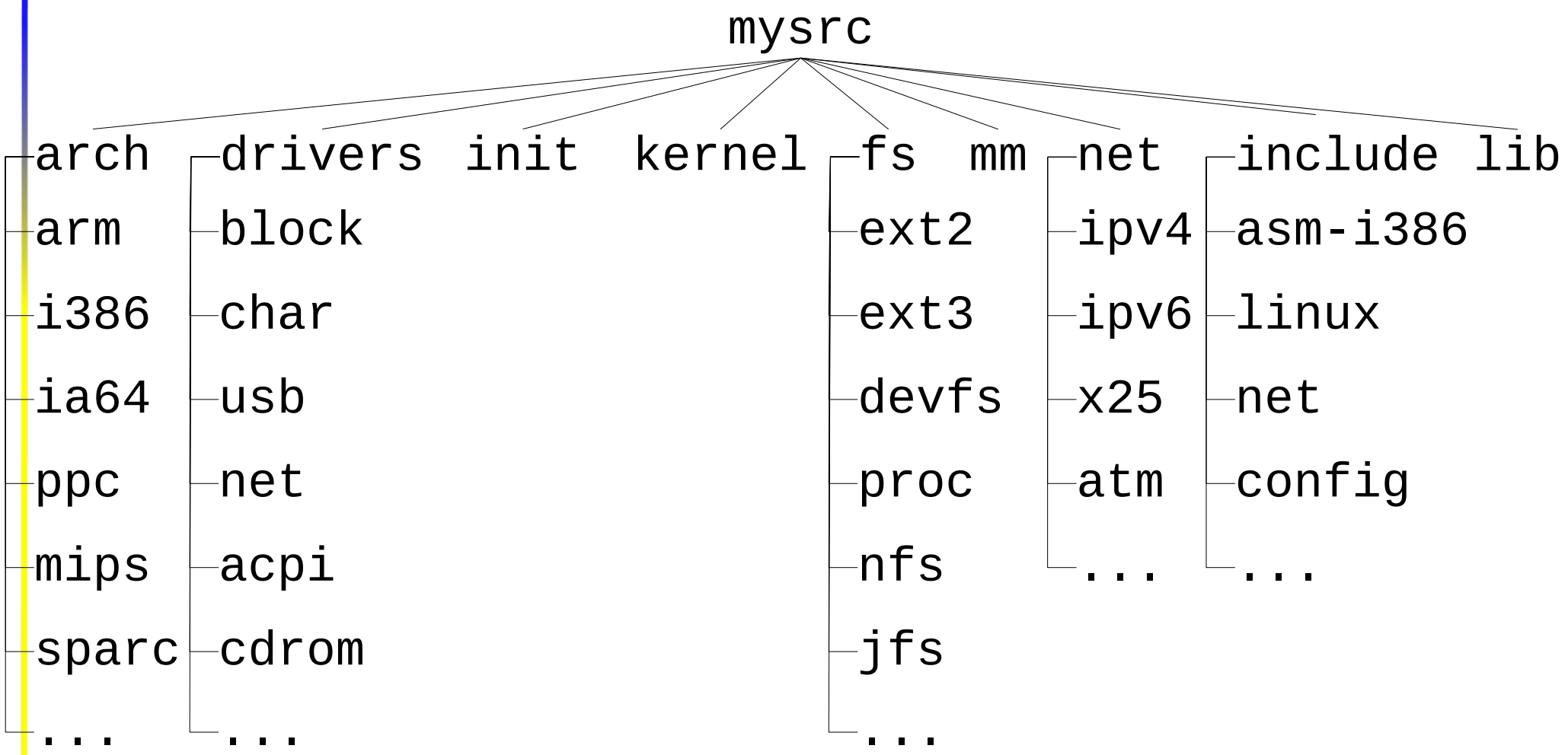
Quelle distribution?

- Quel noyau?
  - Facile: `uname -a` , ou `cat /proc/version`
- Quelle distribution?
  - /etc/redhat-release
  - /etc/fedora-release
  - /etc/slackware-version
  - /etc/debian_version
- On trouve des scripts qui font une recherche sur ces noms

Linux et Temps-réel

- Linux 2.4:
  - Requiert des modifications (patch) RT-preempt, Low Latency, O(1) scheduler
- Linux 2.6:
  - Certaines modifications intégrées dans les sources du noyau
  - Patch PREEMP-RT si nécessaire
- Combiner 2 OS côte à côte!
  - Virtualisation (VirtualLogix, Tenasys, PikeOS, WindRiver, ...)
  - Adeos/Xenomai, RTLinux...

Arborescence des sources d'un noyau Linux



Configuration et compilation

- tar -zxvf mon-archive.tgz
- make config | xconfig |menuconfig
  - On peut aussi importer un fichier de configuration
- make
  - => bzImage
  - => modules
- make install // noyau
- make modules_install

Modules Linux

- Fichier objet (suffixe: .ko)
- Chargeable / déchargeable dynamiquement
  - Selon configuration du noyau
  - Un pilote de périphérique,
  - Un file système,
  - Autre,...
- Commandes (super-utilisateur)
  - lsmod, insmod, rmmod, depmod, modinfo
  - modprobe

Séquence de Boot

- Firmware charge un bootloader: Lilo ou Grub, ou ...
- Grub: Charge le noyau + initrd (ou initramfs)
- Noyau
  - Initialisation, Installation de « initrd »
  - Démarrage de /linuxrc (ou /init)
- Linuxrc (/init)
  - Détermine les modules nécessaires à la poursuite du démarrage: accès au file system racine
  - Monte le file system racine, Pivote les racines

Initrd

- Initrd:
  - File system (ext2 ou autre) stocké en mémoire
  - Construction complexe:
    - Créer un fichier normal (via dd)
    - Initialiser comme un file system: mkfs, mke2fs,...
    - Le monter avec un device loopback (droits super utilisateur)
    - Le peupler avec les modules, fichiers requis
    - Démonter, Compresser
- Nécessite file system correspondant dans le noyau

Initramfs

- Utilisé dans les versions de noyau >= 2.6.13
  - Compatibilité / support initrd assuré(e)
- Basé sur « cpio », utilisateur normal
- Noyau au démarrage
  - Connaît archive de type cpio
  - Transforme archive cpio en file system ramfs
  - Exécute /init
  - Suite similaire à initrd.

Montage racine

- Tous les drivers et file systems inclus dans le noyau,
  - => pas besoin de initrd/initramfs
- Utilisation de initrd
- Utilisation de initramfs

BusyBox: Principes

http://www.busybox.net/

- Créer une commande unique fournissant les services des commandes usuelles
  - Exemples: ls, cp, sed, diff, mkdir...
  - Gains:
    - Réutilisation de code commun non fourni par des librairies
    - Évite les copies bibliothèques statiques sur disque/mémoire, mais on peut utiliser les librairies dynamiques
- Fournir des versions réduites de ces commandes
- Voir aussi FreeBSD `crunchgen`

BusyBox

- On peut exécuter les commandes individuelles:
  - # busybox ls # discrimine sur argv[1]
  - # ln busybox ls; ls # discrimine sur argv[0]
- Le "contenu" de BusyBox est configurable par makefile: `make menuconfig`
- On peut ajouter ses propres "applets" (attention, GPL, pas LGPL)
- Souvent combiné avec µClibc, et LinuxTiny

BusyBox (GPL v2)

- Configuration, génération
  - `make menuconfig, make`
- Compilation croisée
  - `make CROSS_COMPILE=arm-linux-uclibcgnueabi-`
- Linux 2.4 et suivants,
  - Portable sur systèmes Unix
- Processeurs: ceux supportés par gcc
- Librairies: uClibc et glibc

uClibc (LGPL)
http://www.uclibc.org/

- Initialement créée pour supporter uCLinux
  - Machines sans MMU -> sans support mémoire virtuelle
- Utilisable sur toute machine
- Générer avec une chaîne croisée « toolchain »
- On peut aussi utiliser des versions binaires disponibles sur le site.

Toolchain, binutils

(http://sources.redhat.com/binutils/)

- Compilateur gcc + GNU binutils
  - as, ld
  - addr2line - Converts addresses into filenames and line numbers.
  - ar - A utility for creating, modifying and extracting from archives.
  - nm - Lists symbols from object files.
  - objdump - Displays information from object files.
  - ranlib - Generates an index to the contents of an archive.
  - readelf - Displays information from any ELF format object file.
  - size - Lists the section sizes of an object or archive file.
  - strings - Lists printable strings from files.

Produire une chaîne de compilation croisée

- « crosstools » (parmi d'autres)
  - http://www.kegel.com/crosstool/
  - Adapter les fichiers de « demo » à vos besoins (processeur cible, version compilateur...)
- « buildroot » (parmi d'autres)
  - http://buildroot.uclibc.org/
  - Construit une chaîne, mais aussi un noyau Linux et un système de fichiers racine pour une machine cible, basé sur uClibc

OpenWrt

- Distribution Linux embarqué
  - Focus sur intégration réseau
  - Utilisation dans routeurs WiFi par exemple
- Fournit des scripts
  - Télé-chargement des logiciels (Linux, uClibc, Busybox..)
  - Applique des patchs (en fonction de la version)
  - Système de configuration et de paquetage spécifiques

OpenEmbedded

http://www.openembedded.org

- Approche similaire à OpenWrt
- Utilisé par OpenMoko, Palm pour WebOS
- Supporte environnement graphique pour les machines cibles (FB, Qt)
- Permet de maintenir les modifications / adaptations dans des « overlays »

OpenEmbedded

http://www.openembedded.org

- Compilation croisée pour # processeurs
- Construction distribution complète pour mémoires flash
- Possibilité de construire pour d'autres environnements (OpenWrt)
- Multiples format de paquetages (rpm, deb, ipk)
- Librairies: uClibc, glibc, eglibc
- ...