Systèmes et Logiciels Embarqués

UE Programmation système et temps réel

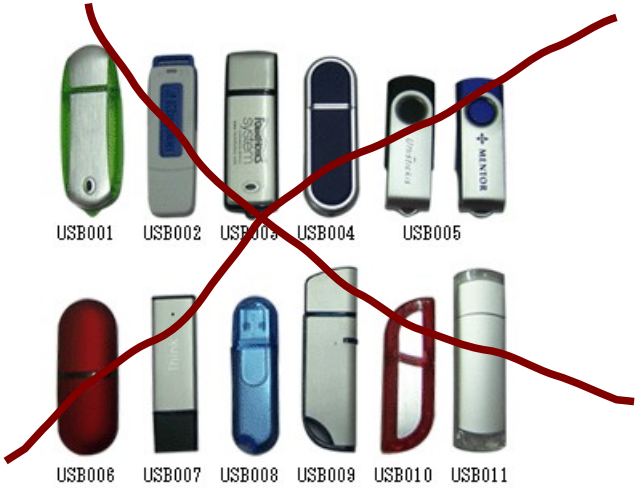
Jean-Patrick Gelas
<jean-patrick.gelas@univ-lyon1.fr>
Université Claude Bernard - Lyon 1

Organisation / Matériel

Durée : 1h CM + 2x3h TP

TP GNU/Linux (Ubuntu, Debian, ...)

Apportez une clé USB !



Fonctionnement

- Pas de soudures, pas de programmation en assembleur.
- Assemblage de ressources logicielles, paramétrage, script shell,...
- Apporter une clé USB (>= 1 GB)

Méthode d'évaluation

100 % : Contrôle continue (QCM)

Sources

- **Linux embarqué**, 4nd édition, *Pierre Ficheux, Éric Bénard*, Ed.Eyrolles.
- http://free-electrons.com/articles
- http://tuxmobil.org/
- http://linuxdevices.com

Objectifs de ce cours

- Panorama de l'existant ;
- Méthodologie à suivre pour construire un système embarqué communiquant complet (matériel, OS, système de fichiers, application(s)).
- Initiation basée sur un système d'exploitation ouvert.

Champs d'applications

(liste non exhaustive)

- contrôle de processus industriels ;
- commande numérique, machines outils ;
- automobile ;
- télécommunication : centraux téléphoniques, téléphones mobiles ;
- réseaux informatiques : routeurs ;
- périphériques informatique : imprimante, photocopieurs ;
- aéronautique et transport en général ;
- systèmes médicaux ;
- Etc... (geekerie, mode, Art,...)

Caractéristiques d'un système embarqué

- **Ciblé** : domaine d'action limité (sauf cas particulier).
- **Fiable et sécurisé** : fonctionnement autonome.
- **Maintenable dans le temps** : durée de vie longue.
- **Spécifique** : interfaces classiques réduites au minimum.
- **Optimisé** : petite taille (*), coût de production, démarrage rapide, temps de réponse, consommation,...

Caractéristiques d'un système embarqué

- **Optimisé** :
  - Taille :
    - Volume physique
    - Mémoires (RAM, stockage)
  - Coût de production,
  - Temps de démarrage,
  - Temps de réponse,
  - Consommation,...

Concepts fondamentaux

- Un logiciel **embarqué** *(embedded software)* est un programme utilisé dans un équipement industriel ou un bien de consommation.
- On parle aussi de logiciel **intégré** ou **dédié** (on achète un produit pour le service qu'il rend et non pas pour le logiciel qu'il embarque).

=> Un équipement est valorisé par son aspect fonctionnel.

Logiciel ou Système embarqué ?

- Un *système embarqué* désigne le plus souvent un système d'exploitation (multi-usage) qui permet :
  - de gérer et partager les ressources matériels ;
  - d'assurer un ensemble de services à travers une interface mieux adaptée.
- Un OS est a priori beaucoup trop complexe et sur-dimensionné, mais...

Avantages de l'utilisation d'un OS

- Il affranchit le développeur d'un travail d'adaptation très proche du matériel ;
- Si l'OS est suffisamment répandu, il permet aux applications industrielles de bénéficier des mêmes avancées technologiques que les applications classiques (TCP/IP, HTTP, FTP,...).
- Un environnement de développement confortable.

Inconvénient *:* Le principal inconvénient de l'utilisation d'un véritable OS est la taille mémoire nécessaire.

L'empreinte mémoire

- L'*empreinte* (*footprint*) est la taille mémoire occupée par le système.
- Les systèmes embarqués doivent avoir une faible empreinte mémoire pour optimiser les **coûts** de production.

=> La réduction de l'empreinte mémoire est la tâche principale d'un développeur de système embarqué.

	Serveur	Desktop	PC emb.	Emb. gros	Emb. moyen	Emb. typique	Profondément enfui
RAM	2 à 64 Go	2 à 16Go	32-512 Mo	8-32Mo	2-8Mo	4 à 0.1	> 100 Ko

Langages utilisés

- Pour des raisons de contraintes matérielles, le langage *assembleur* a longtemps été le langage de prédilection des technologies de l'embarqué (optimisation de la taille du code et des performances).

Les langages C et C++ (standard POSIX) restent aujourd'hui le choix favori des développeurs .

  - Le C permet de rester proche du matériel.
  - Il nécessite d'utiliser le compilateur de la manière la plus efficace possible en utilisant au maximum les options d'optimisation adapté au matériel (ex: GNU C Compiler -O2, -O3, -Os, -m386, -mpentium, etc.)

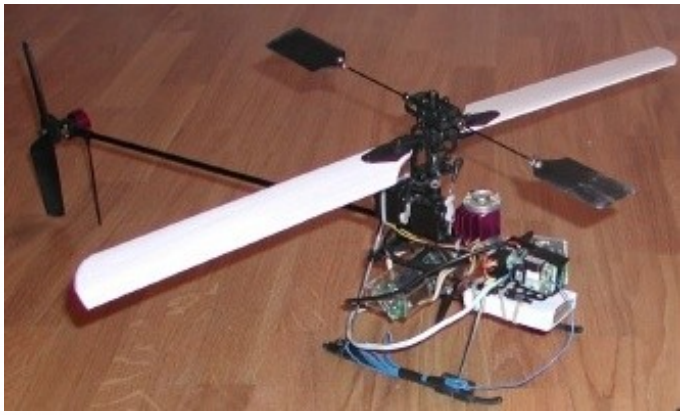
- Langage de script (sur les UNIX-like). Plus facile à maintenir (ex: ash 60Ko, msh 30Ko).

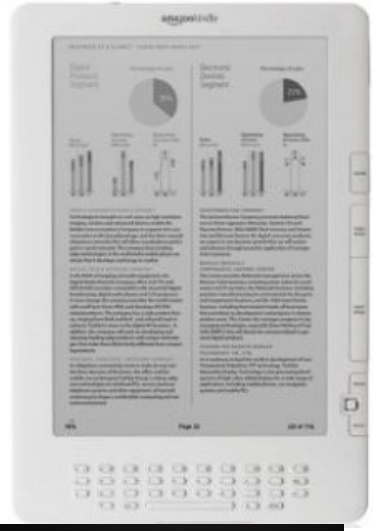
Systèmes existants

(non exhaustif)

- **VxWorks** : noyau RT le plus utilisé dans l'industrie. Supporte TCP/IP et une API socket. Coût des licences important.
- **QNX** : noyau RT de type UNIX. Intègre une interface garphique proche de X Window (Photon). QNX peut être utilisé gratuitement pour des applications non commerciales. Très faible empreinte mémoire.
- **uC/OS** : destiné à des micro-contrôleurs type Motorola 68HC11. Support de TCP/IP. Gratuit pour l'enseignement.
- **Windows CE** : Victime d'une réputation de fiabilité approximative...
- **Lynx OS** : Système RT conforme à la norme POSIX.
- **Nucleus** : Noyau RT avec support de TCP/IP, une interface graphique, un navigateur web et serveur HTTP. Livré avec les sources, pas de royalties à payer pour la redistribution.
- **eCos** : Noyau RT *(Embeddable Configurable OS)*, faible empreinte mémoire, basé sur Linux et la chaîne de *cross-compilation* GNU (POSIX). Support de TCP/IP. Licence proche GPL. Largement utilisé (automobile, imprimante, lecteur MP3). Support des processeurs x86, PowerPC, SH3 Hitachi ou StrongARM.

FROM: MOOBELLA



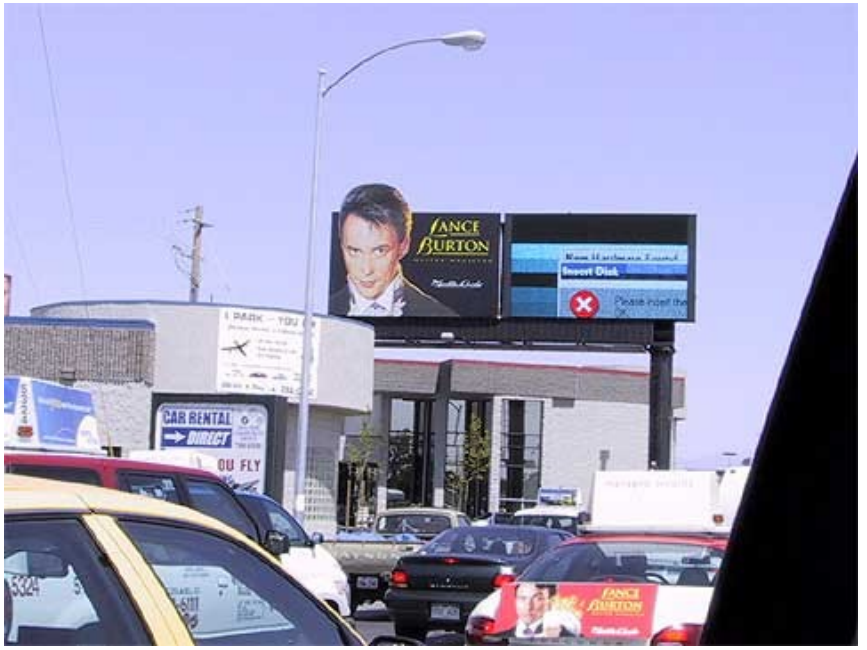


GNU/Linux comme système embarqué : Avantages et contraintes

- Les avantages de l'*Open Source* :
  - redistribution sans royalties ;
  - disponibilité du code source ;
  - possibilité de réaliser un développement dérivé de ce code source.
- Les contraintes :
  - L'*open-source* a une approche communautaire qui peut provoquer la méfiance des décideurs (connotation anti-libérale, anti-profit).
  - Support technique.

Pourquoi GNU/Linux est-il adapté à l'embarqué ?

- Fiabilité
- Faible coût
- Performances
- Portabilité et adaptabilité
- Ouverture



Pourquoi GNU/Linux est-il **in**adapté à l'embarqué ?

- Un noyau Linux occupe au minimum 400kB (compressé). 1MB pour un système minimal fonctionnel. 4MB de RAM minimum.
- Si l'application nécessite uniquement des fonctions de base (ex: pas de réseaux, pas de multitâche) et n'est pas destiné a évoluer.
- Si les contraintes matérielles ne sont pas compatibles.
- L'usage de la GPL/LGPL peut se révéler contraignante.
- La conformité par rapport à certains standards industriels (ex: standards de l'aéronautique) peut être difficile à valider.

Choix matériel pour un système embarqué : Architecture PC ou non ?

- Utilisez une architecture PC pour la phase de maquettage et d'étude de fonctionnalité.
- Linux est disponible sur un grand nombre de processeurs. Le portage d'un noyau Linux vers un nouveau processeur est envisageable.
- **Limitez une étude matérielle a une production importante.**

Choix matériel pour un système embarqué : Processeur : MMU ou non ?

- Linux a été initialement développé sur la base du mécanisme de *mémoire protégée* du processeur Intel 80386.
- Ce mécanisme repose sur un composant matériel appelé MMU (*Memory Management Unit*) :
  - Il interdit a un processus d'écraser l'espace mémoire d'un autre processus (*segmentation violation*).
  - Il réalise la conversion entre les adresses physiques et adresses logiques (vu par le processus et allouées par l'OS).
- Les processeurs équipés de MMU (la majorité) sont plus gourmand en ressources matérielles. Pour les applications « profondément enfouies » *(deeply embedded)* on utilisera des processeurs dépourvus de MMU.
- *uClinux* : Linux sans MMU (pour micro-contrôleurs), http://www.uclinux.org. Portage basé sur la version 2.0, 2.4 et 2.6 du noyau Linux. L'API du système est indentique au vrai noyau Linux (basé sur une libc différent de glibc).

Les processeurs

- Avec MMU compatible x86 :
  Intel (Atom, Pentium, Celeron), AMD (Elan530), National Semiconductor (Geode), VIA (C3), STPC, Transmeta (Crusoe), *etc*.
- Architecture ARM (ARM7, ARM9, Xscale, etc.)
- ATMEL (AT91RM9200 (Ethernet, USB host/device)).
- ...

La mémoire de masse

- Utilisée pour stocker l'image de l'OS et les données lues ou écrites en cours de fonctionnement de ce système.
- Un système embarqué utilise généralement un faible espace de stockage (les distributions Linux embarquées font entre 2 et 10 MB).
- Un disque dur est :
  - inadapté a un environnement mobile (fragile) ;
  - consommateur d'énergie (dissipateur thermique) ;
  - peut générer du bruit.
- Un système embarqué utilise généralement de la mémoire flash *(flash memory)* : DiskOnChip (MTD), CompactFlash (interface IDE), SSD ?...
  - temps d'accès (en écriture) supérieur à celui de la mémoire vive ;
  - la durée de vie fortement influencée par le nombre d'écriture par unité de temps (*wear leveling* chargé de répartir les écritures de manière uniforme).

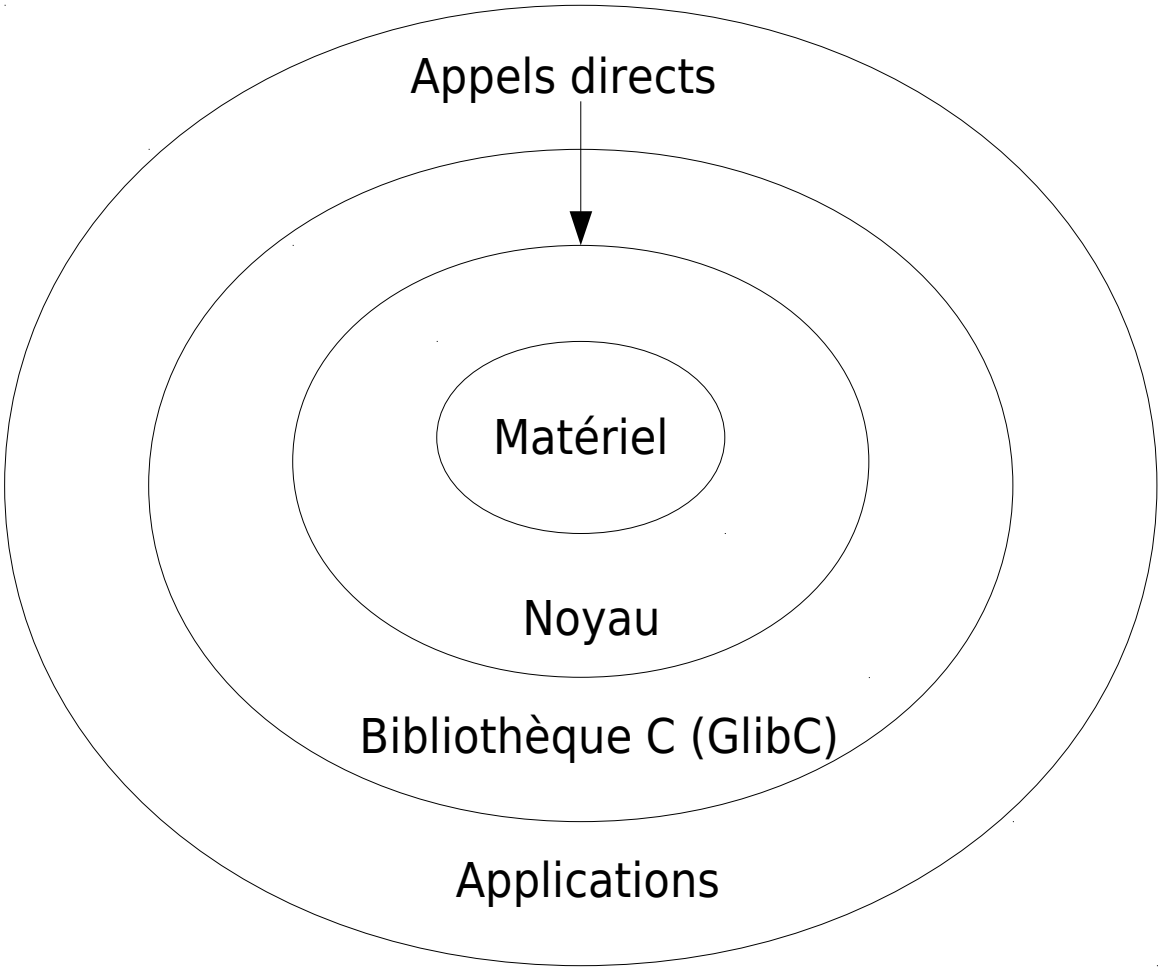
Systèmes de fichiers

- **ext2/ext3/ext4**
  Le système de fichiers *ext3* est l'extension journalisée du système de fichier *ext2* (Labo MASI, Jussieu, Rémy Card).
- **JFFS2** *(Journaling Flash File System, version 2)*
  Lié au projet MTD. Gestion des blocs erronés *(bad blocks).* Récupération d'espace *(Garbage Collector).* Compression des données en temps réel.

Systèmes de fichiers (suite)

- **CRAMFS** (Compressed ROM File System)
  Système de fichiers en lecture seule compressé (zlib). La taille maximale de chaque fichier est limité a 16Mo et la taille totale du système de fichiers à 256Mo.
- **SquashFS**
  Système de fichiers en lecture seule compressé (zlib). Taille maxi des fichiers ou du système de fichiers = 2^64 octets. Taille des blocs jusqu'à 1 Mo (128 Ko par défaut pour un bon taux de compression). Suppression des fichiers dupliqués détectée. Support des archi *big-* et *little-endian*.

Structure du système Linux



Noyau (kernel) : réalise les fonctions essentielles (gestion des tâches et mémoire), interface entre le matériel et les applications (pilotes).

libc : bibliothèque principale contenant les fonctions de base utilisées par les applicatifs.

Applications (ou commandes) : livrées avec le système ou développées pour des besoins spécifiques.

Schéma de démarrage d'un système Linux

1) POST (*Power-On Self Test*)

2) *Bootstrap* ou **bootloader** : Le bios d'un PC se charge de lire le MBR (*Master Boot Record*, 512 octets) à partir de son lecteur de boot (hd, fd, CD, usb,...).

3)Second-stage *boot loader* : LILO, GRUB, *syslinux*,...

4) Chargement du noyau Linux (ex: /boot/vmlinuz-2.6.xx-yy): initialisation des périphériques matériels. Chargement de sa partition principal (*root partition*).

5) Exécution du programme */sbin/init* (`/etc/inittab`) ou */bin/systemd*

6) Lancement des services...

Disques mémoire
(Ramdisks ou tmpfs)

- Linux offre la possibilité de travailler sur des disques mémoire.
  - rapidité d'accès aux données ;
  - faible coût de la RAM par rapport à d'autres supports physiques comme la mémoire flash.
- Nécessite de valider le support dans le noyau (Menu *Block devices*).
- Chaque disque mémoire est vu à travers un pilote de type *bloc* /dev/ramX (0<=X<=20).

```
# dd if=/dev/zero of=/dev/ram2 bs=1k count=2048
# mke2fs /dev/ram2
# mount -t ext2 /dev/ram2 /mnt/tmp
```

```
mount -t tmpfs -o size=256M tmpfs
/media/montmpfs
```

Quelques fichiers utiles

- /etc/
  - fstab
  - inittab
  - network/... ou configsys/...
  - hostname, hosts, **resolv.conf**
  - init.d/...
- /sbin/
  - init

Quelques répertoires particuliers /dev, /proc, /var

- **/dev** : On y trouve les pseudo-fichiers ou noeuds (qui ne consomme qu'un *i-node*). Ces fichiers servent de canaux de communication avec les périphériques (disques, cartes son, etc.).

- **/proc et /sys**: sont des systèmes de fichiers virtuels, utilisés pour communiquer avec l'espace utilisateur. Un pilote de périphérique peut y ajouter dynamiquement des fichiers et répertoires lors de chargement de modules associés. Il est utilisable en lecture et écriture.
- **/var** : contient des fichiers et sous-répertoire dont l'encombrement peut augmenter fortement au cours du fonctionnement (idem **/tmp**). Il vaut mieux le mettre sur une partition dédiée. On l'utilise pour stocker :
  - les fichiers de fonctionnement (pid files) ;
  - les fichiers verrous (lock files) ;
  - les fichiers de trace (log files).

initrd

initial ramdisk

- Image d'un système de fichiers (temporaire) utilisé par Linux durant le boot.
- L'image est chargé en mémoire vive et permet d'avoir un système de fichiers (minimal).
- Généralement utilisé pour la préparation avant que le vrai système de fichiers racine puisse être monté ou bien démarrer des systèmes sans disque *(diskless).*

Quelques commandes utiles

- **fdisk**, cfdisk, sfdisk,...
- hdparm, **mkfs.***xxx*, df, tune2fs (ext2/3)
- insmod, lsmod, modprobe, **dmesg**
- **file**, **ldd**, strace, nm, string, **strip**
- tcpdump, wireshark, ping, traceroute,...
- **syslinux**, lilo, **grub**
- **chroot**, pivot_root
- mount : mount -t iso9660 -o loop=**/dev/loop0** imgcd.iso /mnt/cdrom

QEMU

- Emulateur de système ou machine virtuelle (Open source).
- CPU supportés : x86, ARM, Sparc, PowerPC, Mips, m68k (Coldfire),...
- Matériel émulé (x86) :
  - i440FX host PCI bridge et PIIX3 PCI to ISA bridge
  - Cirrus CLGD 5446 PCI VGA card ou une simple carte VGA avec les extensions VESA de Bochs VESA
  - Clavier et souris PS/2
  - 2 interfaces IDE PCI avec support des disques durs et CD-ROM
  - Lecteur de disquette
  - Carte réseau NE2000 PCI
  - Ports série
  - Carte son Soundblaster 16
  - BIOS du projet Bochs et le BIOS VGA des projets Plex86/Bochs
- Émule en permanence le processeur.
  Réduction des performances (facteur de 5 à 10).

----------THE END----------

TP1 - Échauffement

Diagnostiquer la mémoire d'un système x86.

- URL : http://perso.univ-lyon1.fr/jean-patrick.gelas
  (section Embarque)
- Commandes utiles à connaître:
  - **dd** et ses options *if, of, bs, count* ;
  - file ;
  - mkfs.dos ;
  - qemu

TP2 - Élagage

Réduire un système de fichiers au strict minimum

- URL : http://perso.univ-lyon1.fr/jean-patrick.gelas
  (section Embarque)
- Le noyau et le système de fichiers initial sont donnés.
- Utilisation d'*initrd*.
- Tests avec Qemu

  qemu -kernel *vmlinuz* **-initrd** *inird.img* **-hda** ***/dev/zero***