ENSEIRB - 4JLG

Linux embarqué, Linux Temps Réel : présentation

4èmes Journées Linux de Grenoble
Linux Embarqué et Temps Réel
27 - 28 Novembre 2003

Patrice KADIONIK

email : kadionik@enseirb.fr
http : http://www.enseirb.fr/~kadionik

INTRODUCTION

- Présentation des solutions techniques pour la mise en œuvre de Linux dans l ’univers des systèmes embarqués :
  - Linux embarqué : Les concepts. L ’ offre aujourd ’hui.
  - Le Temps Réel et Linux. Les concepts. L ’ offre aujourd ’hui.

- Pour pouvoir ensuite se poser les bonnes questions avant d ’opérer le bon choix.

CHAPITRE 1 :
LINUX EMBARQUE :
LINUX POUR L ’EMBARQUE

PARTIE 1 :
LE PHENOMENE LINUX EMBARQUE

LES SYSTEMES EMBARQUES ET LINUX

- Linux depuis presque 3 ans est en train de conquérir un domaine où on ne l'attendait pas vraiment : l'univers des systèmes embarqués.

- Pourquoi retrouve-t-on Linux dans l'embarqué ? Tout d'abord pour ses qualités qu'on lui reconnaît maintenant dans l'environnement plus standard du PC grand public :
  - Libre, disponible gratuitement au niveau source : pas de royalties à reverser.
  - Ouvert.
  - Différentes distributions proposées pour coller au mieux à un type d'application.

LES SYSTEMES EMBARQUES ET LINUX

- Pourquoi retrouve-t-on Linux dans l'embarqué ? Tout d'abord pour ses qualités qu'on lui reconnaît maintenant dans l'environnement plus standard du PC grand public :
  - Stable et efficace.
  - Aide rapide en cas de problèmes par la communauté Internet des développeurs Linux.
  - Nombre de plus en plus important de logiciels disponibles.
  - **Connectivité IP en standard**.

LES SYSTEMES EMBARQUES ET LINUX

- Linux a aussi d'autres atouts très importants pour les systèmes embarqués :
  - Portage sur processeurs autres que x86 : PowerPC, ARM, MIPS, 68K, ColdFire…
  - Taille du noyau modeste compatible avec les tailles de mémoires utilisées dans un système embarqué (<500 Ko).
  - Différentes distributions proposées suivant le domaine : routeur IP, PDA, téléphone…
  - Support du chargement dynamique de modules qui permet d'optimiser la taille du noyau.
  - Migration rapide et en douceur pour un spécialiste Linux à Linux embarqué ; ce qui réduit les temps de formation (et les coûts…).

LES SYSTEMES EMBARQUES ET LINUX

- On a en fait entendu parler pour la première fois officiellement de Linux embarqué à une exposition *Linux World* en 1999 où les sociétés Motorola, Force et Ziatech ont présenté un système CompactPCI fonctionnant sous Linux.

- En 2000 a été créé le consortium Linux embarqué (*Embedded Linux Consortium*) dont le but est de centraliser et de promouvoir les développements de solutions Linux embarqué. Ce consortium regroupe des éditeurs de distribution Linux, des éditeurs de systèmes Temps Réel propriétaires (comme WindRiver pour VxWorks) et des fabricants de composants. Il compte actuellement plus de 100 membres.

LES SYSTEMES EMBARQUES ET LINUX

- Les distributions Linux embarqué ont une part de marché grandissante face à des distributions propriétaires généralement Temps Réel comme VxWorks, pSOS, QNX… où l'on est d'abord obligé de payer pour accéder à la plateforme de développement puis de payer des royalties pour chaque système (ou cible) que l'on commercialise ensuite.

- Il est à noter que l'on observe une évolution de ce système à péage de certains face à la " menace " Linux.

LINUX EMBARQUE

- Linux embarqué est une adaptation du noyau Linux à un système embarqué. Suivant les capacités du système, on retrouve tout ou partie des fonctionnalités du noyau :
  - Moins de services disponibles.
  - Moins de mémoire requise (< 8 Mo).
  - Boot depuis une mémoire ROM (FLASH).
  - Pas de clavier ou de souris requis.
  - Logiciels spéciaux pour piloter les périphériques du système (écran LCD, flash disk, Disk On Chip DOC, touch screen…).

LINUX EMBARQUE

- Une version de Linux embarqué peut être spécialement configurée pour coller à une plateforme ou application précise :
  - Linux embarqué pour routeur IP.
  - Linux embarqué sur PDA.
  - Linux embarqué pour microcontrôleur sans MMU.
  - Linux embarqué sur processeur 80286 et inférieur.
  - ...

OUTILS POUR LINUX EMBARQUE

- On utilise pour le développement sous Linux embarqué les outils traditionnels GNU :
  - (cross) compilateurs C/C++. C est préférable pour limiter la taille des exécutables.
  - IDE.
  - Debugger (GDB).
  - Simulateur.

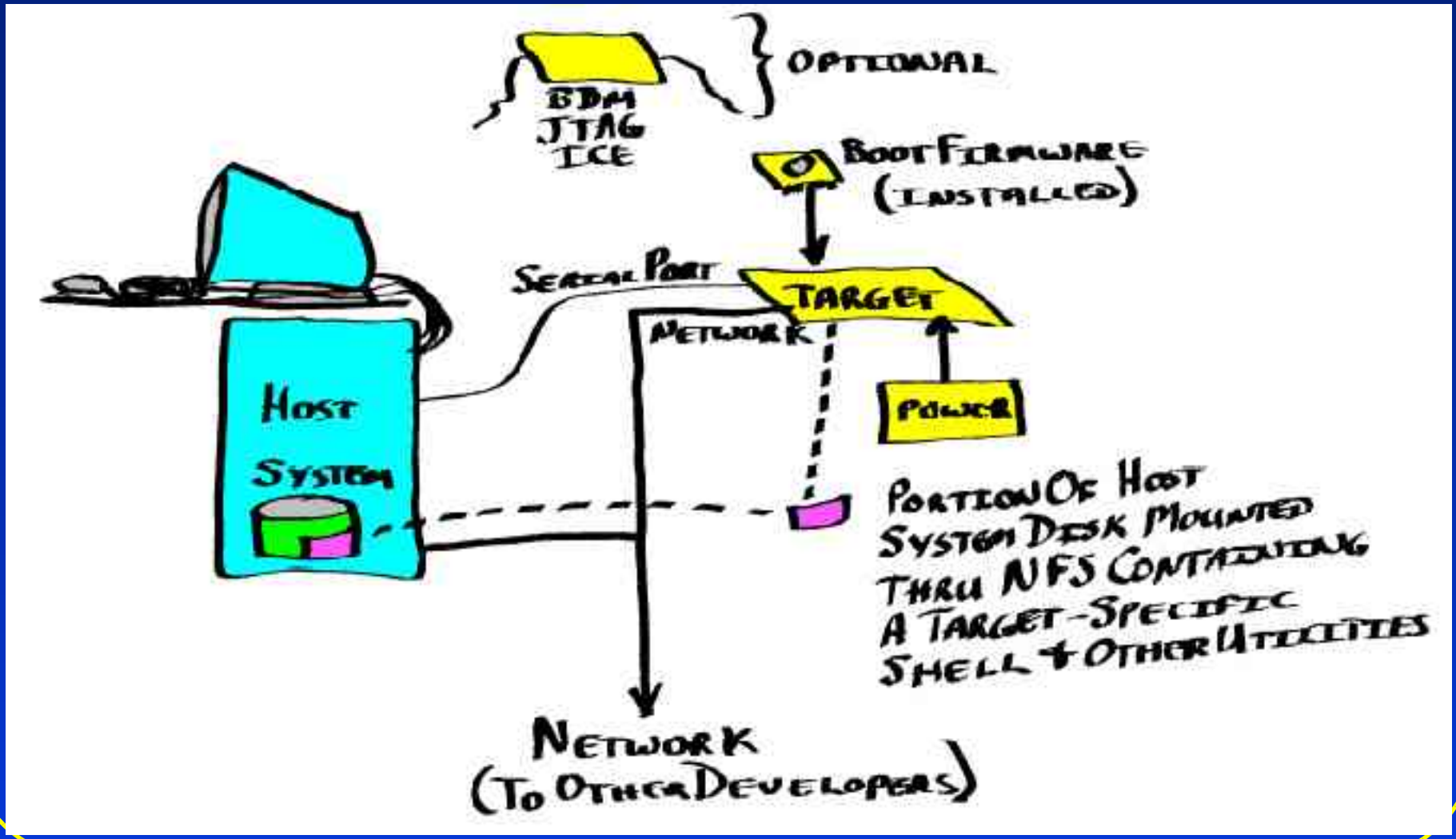
OUTILS POUR LINUX EMBARQUE

- On utilise pour le développement sous Linux embarqué un PC de développement sous Linux (l 'hôte) avec une chaîne de compilation croisée en fonction du processeur embarqué sur le système (la cible).

- L 'exécutable ainsi produit est téléchargé dans la cible pour pouvoir y être testé. On utilisera alors GDB pour débugger l 'application par le réseau que l 'on pourra coupler avec une interface graphique de type DDD (Data Display Debugger).

- Un montage NFS depuis la cible d 'un répertoire du PC hôte permet de simplifier la phase de téléchargement.

OUTILS POUR LINUX EMBARQUE



OUTILS POUR LINUX EMBARQUE

- Il existe des simulateurs tournant sur le PC hôte pour simuler la cible :
  - Simulateur pour émuler une grande marque de pocket PC.

- Il est possible d 'utiliser d 'émuler complètement un système sur le PC hôte en utilisant le projet UML (*User Mode Linux*). UML permet de créer une machine virtuelle tournant un Linux embarqué correspondant à la cible et à son type de processeur. Cela permet alors de compiler une application directement en natif si l 'on se connecte à cette machine virtuelle…

http://user-mode-linux.sourceforge.net/

OUTILS POUR LINUX EMBARQUE

- Java est aussi supporté.

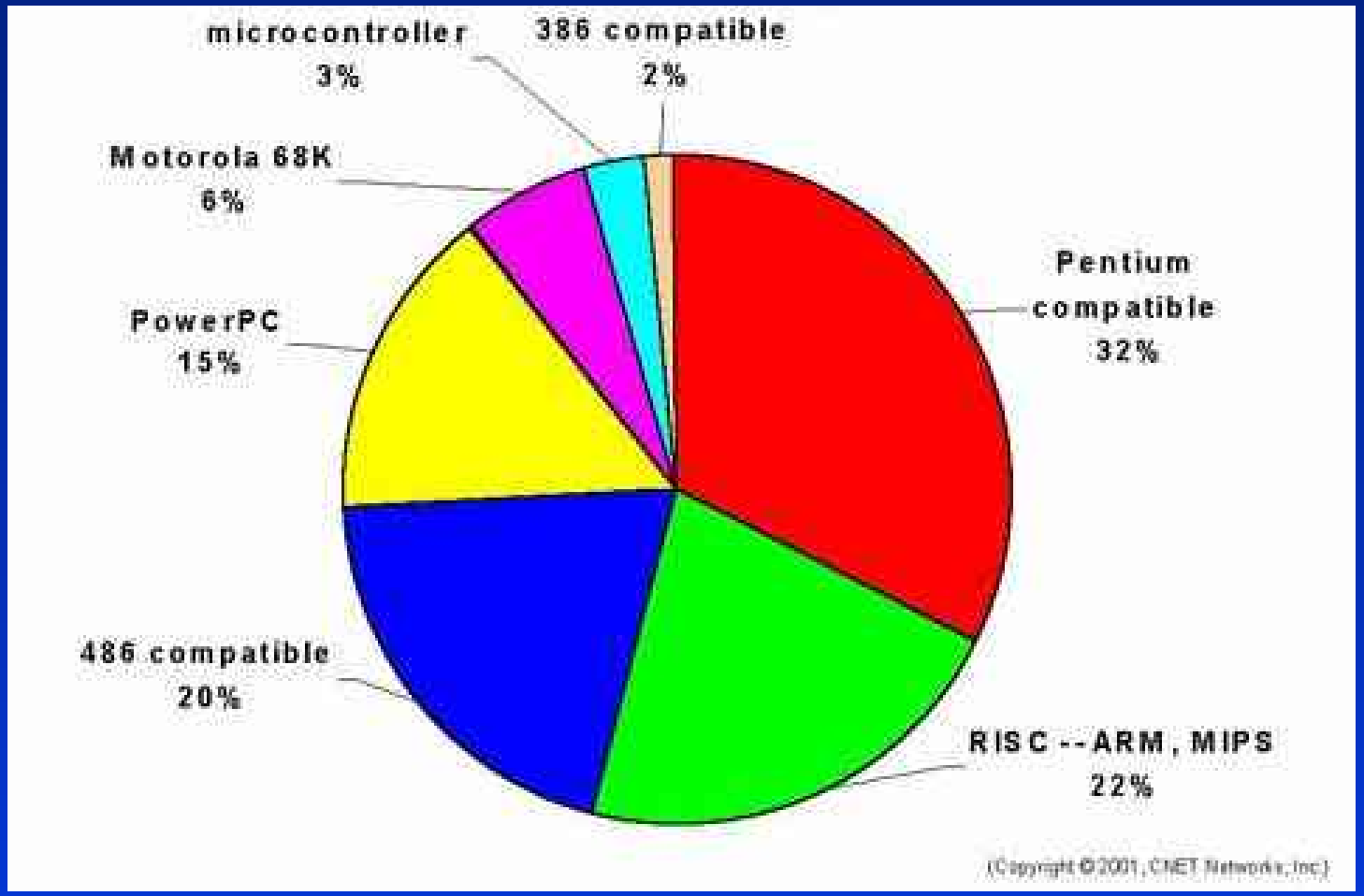
- Il est possible aussi d ’utiliser des interfaces graphiques légères :
  - Microwindows.
  - Nano-X
  - Qt Embedded de Troltech (et dérivés Qtopia, OPIE).
  - Frame buffer
  - ...

LE CHOIX D 'UN PROCESSEUR POUR L 'EMBARQUE

Besoin	Miniature	Petit	Moyen	Haut de gamme	PC embarqué	Embarqué haute disponibilité
Taille RAM	<0,1 Mo	0,1-4 Mo	2-8 Mo	8-32 Mo	16-64 Mo	> x Mo
Taille ROM/FLASH	0,1-0,5 Mo	0,5-2 Mo	2-4 Mo FLASH	4-16 Mo FLASH	xx Mo	Go-To
Processeurs	DragonBall 68K Mcore ColdFire ARM		MIPS Hitachi SH x86 PowerPC			Pentium PowerPC
Caractéristiques matérielles	MMU optionnelle		Ardoise Internet Carte unité centrale System on Chip (*SoC*)			CompactPCI
Exemples d'applications	Caméra numérique PDA Téléphone		Routeur Décodeur Stockage en réseau Imprimante en réseau			Commutateur téléphonique Routeur haute performance Serveur central

- Choix suivant puissance de calcul, taille mémoire...

CHOIX DU PROCESSEUR POUR LINUX EMBARQUE



PROCESSEURS SUPPORTES POUR LINUX EMBARQUE

- Cela dépend essentiellement de la distribution Linux embarqué :

- Par exemple, MontaVista supporte :
  - Intel (x86).
  - PowerPC.
  - MIPS.
  - StrongARM.
  - Hitachi Super-H.

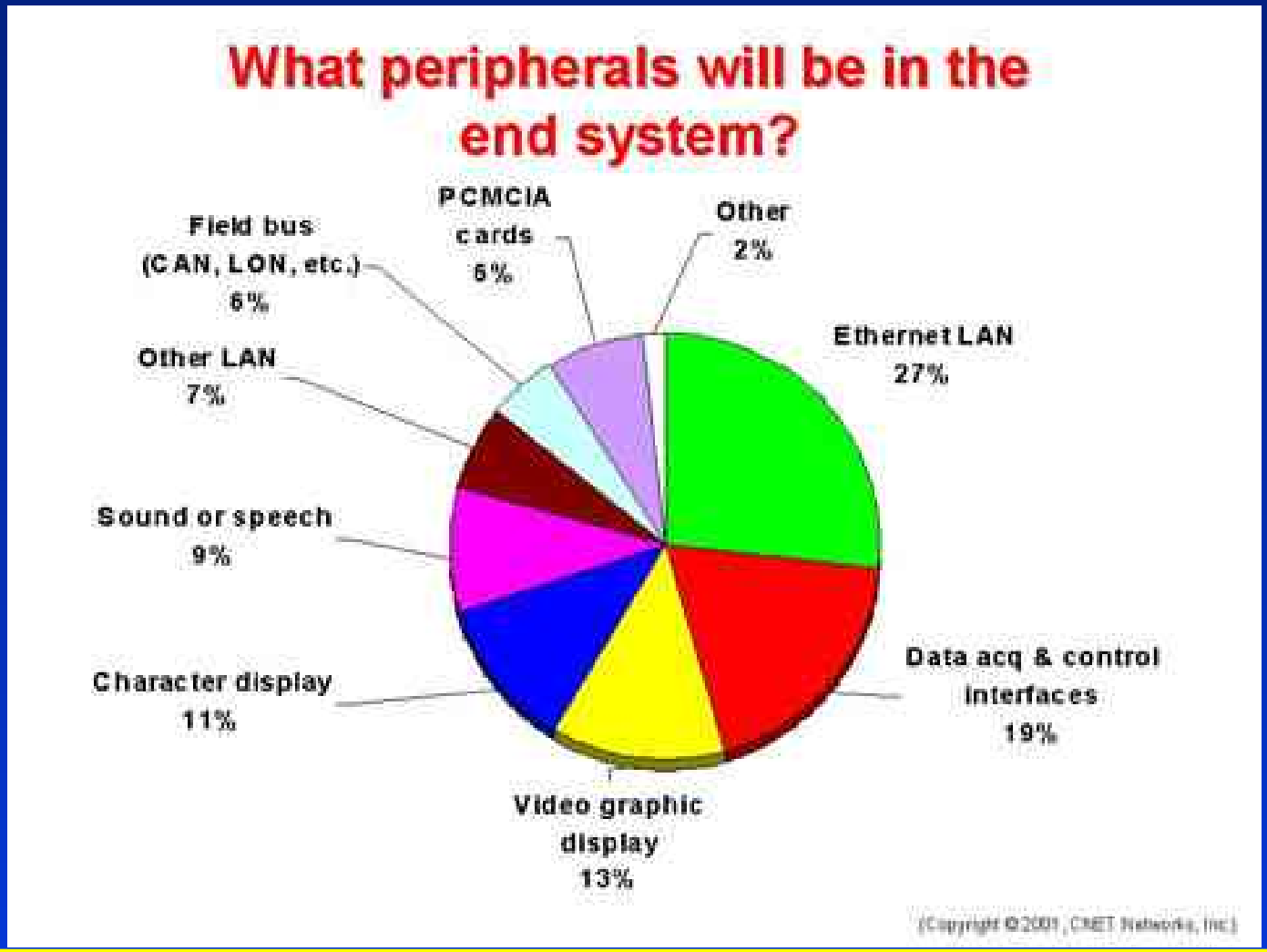
http://www.mvista.com/products/hardware.html

PROCESSEURS SUPPORTES POUR LINUX EMBARQUE

- Par exemple, Lineo/Metrowerks/ Motorola supporte :
  - x86.
  - PowerPC.
  - StrongARM.
  - **Motorola 683xx et ColdFire.** (Lineo était à l'origine du projet µClinux)

http://www.metrowerks.com/embedded/

PERIPHERIQUES POUR LINUX EMBARQUE



CARTES POUR LINUX EMBARQUE

- Little Board (5.75 x 8.0 in.) -- complete systems on a single compact board, expandable with plug-on function modules
- ISA "slot boards" (full-length, 13.8 x 4.8 in.; half-length, 7.1 x 4.8 in.) -- IBM PC plug-in cards which could function as standalone SBCs backplanes)
- PC/104 modules (3.6 x 3.8 in.) -- compact, rugged, self-stacking modules featuring a reliable pin-and-socket board-to-board expansion bus

CARTES POUR LINUX EMBARQUE

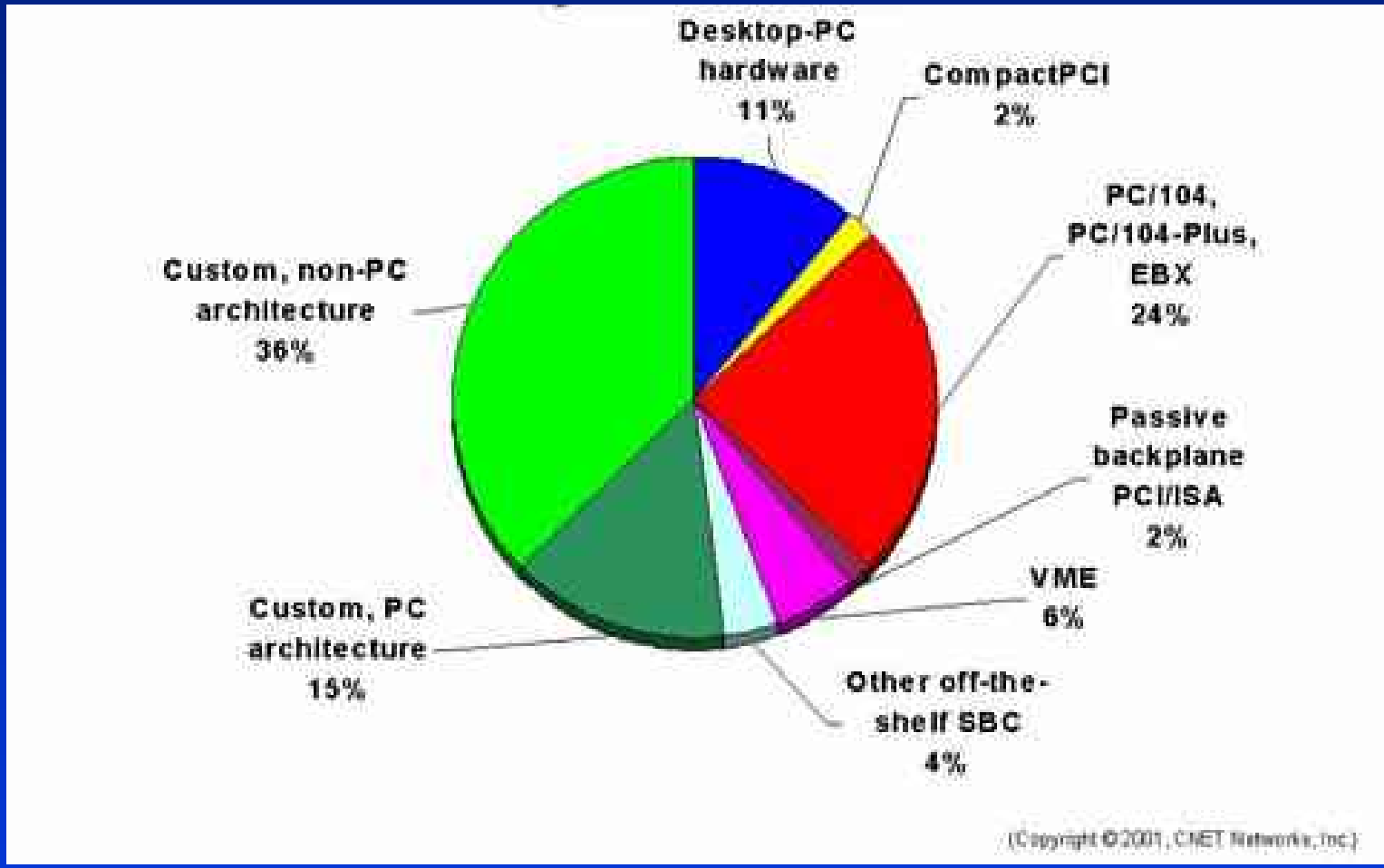
- Bus PCI en plus :

  PC/104-Plus -- PCI added to PC/104

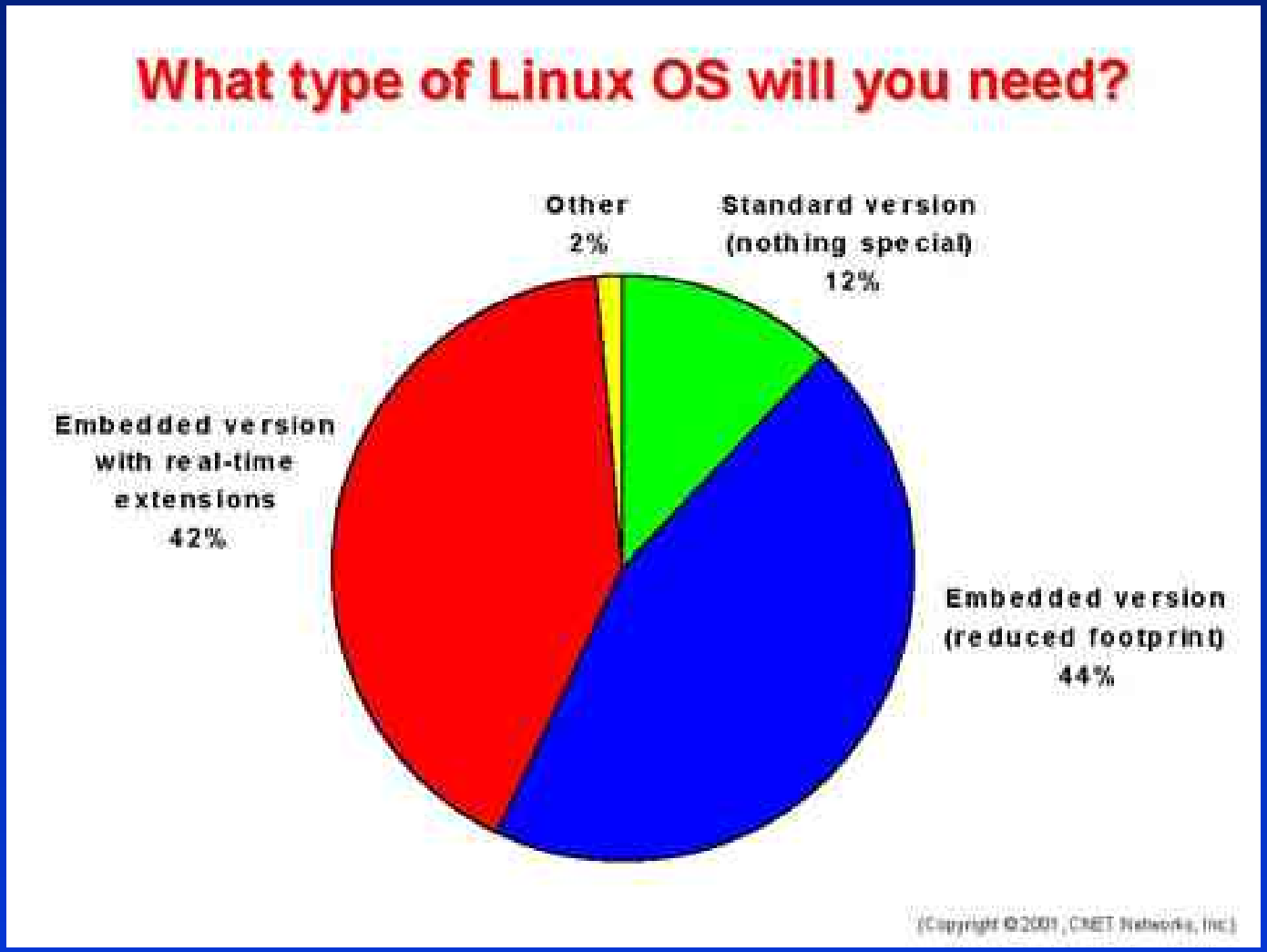
  EBX -- PC/104-Plus added to Little Board

- Cartes au format industriel VME, VXI, PXI...

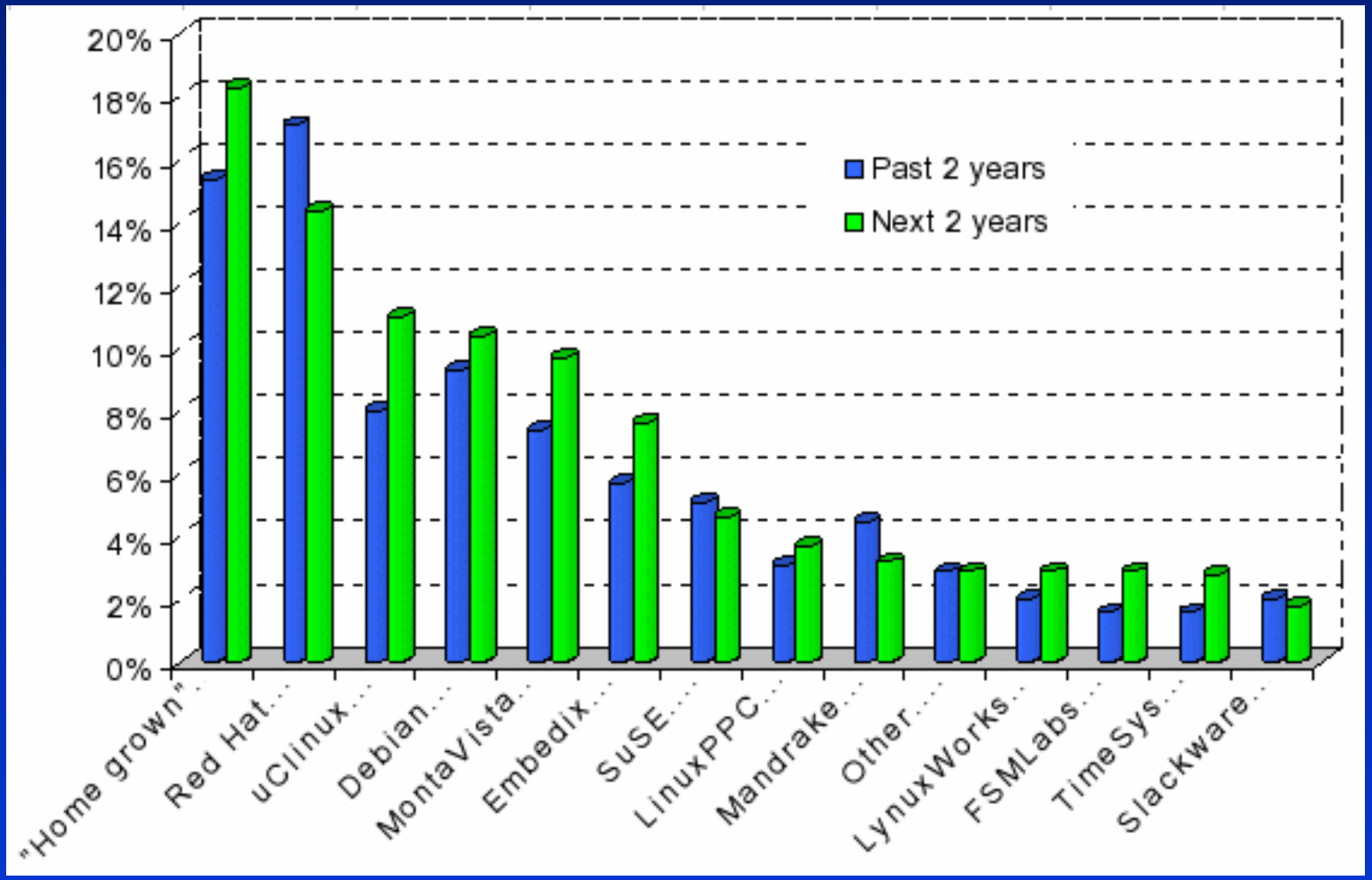
FORMAT DES CARTES CHOISI POUR LINUX EMBARQUE



CHOIX D ’UNE VERSION LINUX EMBARQUE



CHOIX D ’UNE VERSION LINUX EMBARQUE



- Enquête linuxdevices.com juin 2003

POINTS FAIBLES DE LINUX EMBARQUE

- Les drivers Linux pour un périphérique donné ne sont pas toujours disponibles.
- Le manque de standards : GUI, extensions Temps Réel…
- Le manque d'une cohérence marketing.
- Le manque d'outils de qualification d'un système sous Linux (tests de conformité de l'API POSIX pour le Temps Réel ?).
- Le modèle de la licence GPL mal compris (droits et surtout obligations).

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- On pourrait être effrayé de prime abord par la multitude de l'offre Linux embarqué pour faire un choix correspondant à son besoin (s'il est bien défini !).

- Le consortium ELC (Embedded Linux Consortium) a proposé un document de standardisation des applications Linux embarqué.

- Le document ELC Platform Specification (ELCPS) de propose de définir les environnements de programmation basé sur un système Linux embarqué.

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- Le document ELCPS est basé sur un ensemble de standards :
  - The Linux Standard Base 1.2 (LSB)
  - IEEE POSIX 1003.1-2001
  - The Single UNIX Specification v3

- Le document ELCPS de propose de promouvoir :
  - Le développement de systèmes et d'applications Linux embarqué
  - La portabilité des applications

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- Le document ELCPS définit 3 types d'environnements système :
  - 1. Environnement système minimal : pas de stockage de masse, pas d'interaction, profondément enfoui. Monoprocessus.
  - 2. Environnement système intermédiaire : stockage de masse (donc système(s) de fichiers). Cela peut être aussi des systèmes de fichiers en mémoire FLASH. Multiprocessus.
  - 3. Environnement système complet : système général, support réseau, GUI...

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- En fonction de ces 3 environnements, des groupes d'appels système de l'API Linux sont inutiles, obligatoires, optionnels:
  - ELC_C_LANG_MATH : acos(), cos()…
  - ELC_DEVICE_IO : getc(), fflush(), puts(), read()…

- Exemple : IPC obligatoire pour un système moyen ou complet.

- Le document ELCPS est disponible sur le site de ELC http://www.embedded-linux.org/platform.php3

PARTIE 2 :
LES OFFRES LINUX EMBARQUE

LES OFFRES LINUX EMBARQUE

- Les offres de version de Linux embarqué (et Temps Réel) peuvent être rangées dans l'une des 3 catégories suivantes :
  - Les distributions Linux classiques : RedHat , Mandrake, Caldera, Debian, Slackware, Suse…

- Suivant la quantité de mémoire "disque" du système embarqué, il est possible d'édulcorer une distribution classique (<100-150 Mo). Cela tient dans une mémoire Compact Flash (512 Mo…).

- Le projet LFS (Linux From Scratch) explique comment construire son Linux pas à pas depuis rien suivant ses besoins : http://www.linuxfromscratch.org/

LES OFFRES LINUX EMBARQUE

- Les offres de version de Linux embarqué (et Temps Réel) peuvent être rangées dans l'une des 3 catégories suivantes :
  - Les distributions Linux embarqué commerciales :
    - non TR : Montavista/Professional or Carrier Grade or Consumer Electronics Edition (ex Hard Hat Linux), Lineo-Metrowerks-Motorola/Creation Suite for Linux (ex Embeddix), LynuxWorks/BlueCat, RedHat/Embedded
    - TR : FSMLabs/RTLinux Pro, Montavista/ Professional or Carrier Grade or Consumer Electronics Edition (ex Hard Hat Linux), LynuxWorks/BlueCat RT, TimeSys/Linux RTOS Professional or Standard Edition, Lineo-Metrowerks-Motorola/Creation Suite for Linux (ex Embeddix)
    - autres : REDSonic…

LES OFFRES LINUX EMBARQUE

- Les offres de version de Linux embarqué (et Temps Réel) peuvent être rangées dans l'une des 3 catégories suivantes :
  - Les distributions Linux embarqué libres :
    - non TR : µClinux, Embedded Debian Project, PeeWeeLinux, Embedded Linux Workshop (ELW)
    - TR : FSMLabs/RTLinux/free (ex OpenRTLinux GPL), RTAI
    - autres : ADEOS, KURT (TR), Linux-SRT (TR), patchs low latency sur noyau standard (Temps Réel mou), eCOS (TR), ELKS, LEM, LOAF, LRP, Freesco...

LES OFFRES LINUX EMBARQUE

- Voir une liste exhaustive à :
  - http://www.linuxdevices.com/articles/AT9952405558.html
  - http://www.linuxdevices.com/articles/AT8073314981.html

LINUX EMBARQUE COMMERCIAL

- MontaVista/Professional or Carrier Grade or Consumer Electronics Edition :
  - Solution générale (et TR) pour l'embarqué
  - http://www.mvista.com/
  - kit d 'évaluation disponible (preview kit)

- MontaVista Linux Professional Edition
- MontaVista Linux Carrier Grade Edition
- MontaVista Linux Consumer Electronics Edition

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de MontaVista/Professional Edition :
- Board Hardware Support
  - Support for over seventy popular COTS, Evaluation, and Reference boards
  - Support for seven target CPU families with more than 25 CPU variants

- MontaVista Development Environment
  - KDevelop IDE
  - MontaVista Target Configuration Tool
  - MontaVista Library Optimizer Tool
  - Graphical binary and source-level debug
  - Graphical kernel configuration tool
  - Kernel debug (KGDB and hardware debuggers)
  - File system populator

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de MontaVista/Professional Edition :
- Real-time Support
  - MontaVista Linux Preemptible Kernel
  - MontaVista Linux Real-time Scheduler with up to 128 levels of priority

- Rich Networking
  - Extensive complement of clients and servers
  - Rich support for the TCP/IP Suite
  - Broad support for routing, security, tunneling
  - cPCI backplane networking

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de MontaVista/Professional Edition :
- File Systems
  - Conventional and Journaling Filesystems
  - Disk, flash and network-based options

- Development Hosts
  - Linux (Red Hat, Mandrake, SuSE)
  - Solaris 7.0, 8.0
  - Windows 2000/XP (command-line and VMWare)

LINUX EMBARQUE COMMERCIAL

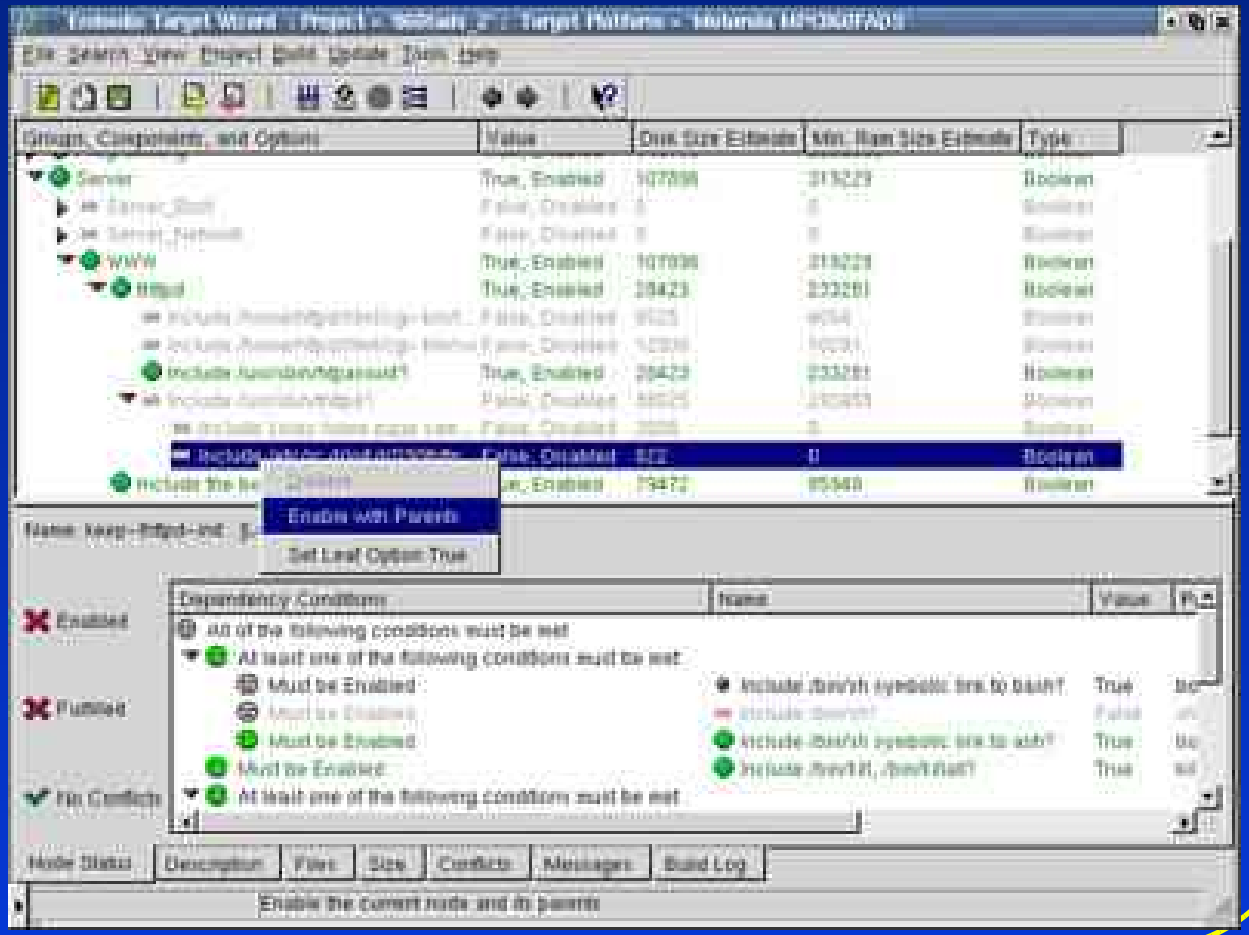
- Lineo-Metrowerks-Motorola/Creation Suite for Linux :
  - http://www.metrowerks.com/
  - kit d ’évaluation disponible

- Caractéristiques de Metrowerks Platform Creation Suite for Linux
- Full-featured and integrated embedded developer tool suite targeting multiple processor families for Linux operating system development.
- Target Wizard Configure, build and deploy
- Package Editor Import open source or binary components
- Linux Kernel Import Tool (LKIT) Import a new linux kernel
- Debian Binary Import Tool (DBIT) Extend embedded linux with a full desktop solution
- CodeWarrior IDE Linux hosted IDE environment

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de Metrowerks Platform Creation Suite for Linux

- The Target Wizard Tools



metrowerks

LINUX EMBARQUE COMMERCIAL

- LynuxWorks/BlueCat :
  - Solution générale pour l'embarqué
  - http://www.bluecat.com/
  - kit d 'évaluation disponible (cible x86)

- Caractéristiques de LynuxWorks/BlueCat :
- Packages that are tailored to your varying requirements for tools and technical support
- A comprehensive set of tools and board support packages for developing, debugging and deploying Linux into embedded environments
- Based on the Linux 2.4.18 kernel, BlueCat Linux scales from small consumer-type devices to large-scale, multi-CPU systems.

LINUX EMBARQUE OPEN SOURCE

- µClinux :
  - Pour processeur 32 bits **sans MMU**.
  - http://www.uclinux.org

- Caractéristiques de µClinux :

•Lineo's uClinux is the ideal OS for non-MMU microprocessors and high-volume embedded systems featuring posix-4, real-time functions, and TCP/IP. uClinux includes a complete TCP/IP stack supporting Ethernet, PPP and SLIP as well as many wireless protocols. uClinux is perfect for remote sensing, monitoring and control applications. And, because uClinux is an open source product, you will never be stuck on a dead end development path.

LINUX EMBARQUE OPEN SOURCE

- Embedded Debian Project :
  - Outil de génération d'un Linux embarqué (OS+FS).
  - http://www.emdebian.org/

- Caractéristiques de Embedded Debian Project :

• EmDebSys a system for the configuration and generation of both a Linux kernel *AND* an operating system (i.e. root filesystem). EmDebSys is being designed to assist embedded Linux developers in configuring and generating small (1 to 10Mb) Linux target systems (ARM, PowerPC, SPARC, Intel x86, Alpha and Motorola 680x0).

LINUX EMBARQUE OPEN SOURCE

- PeeWee Linux :
  - Outil de génération d'un Linux embarqué (OS+FS).
  - http://peeweelinux.com/

- Caractéristiques de PeeWee Linux :

• PeeWeeLinux is an ongoing development effort to provide an environment that makes the configuration and installation of a Linux operating system on an embedded platform as easy and painless as possible.

- Projet similaire Embedded Linux Workshop ELW :
  - Outil de génération d'un Linux embarqué (OS+FS).
  - http://elw.sourceforge.net/

LINUX EMBARQUE OPEN SOURCE SUR DISQUETTE(S)

- Tom's Boot Root :
  - http://www.toms.net/~toehser/rb/
    - Boot/root rescue/emergency floppy image with more stuff than can fit. Bzip2, 1722Mb formatting, and tight compilation options helped jam a lot on. It is useful for "learn unix on a floppy" as it runs from ramdisk, includes the man-pages for everything, and behaves in a generally predictable way.

- Linux Router Project :
  - http://www.linuxrouter.org/
    - LRP is small enough to fit on a single 1.44MB floppy disk, and makes building and maintaining routers, access servers, thin servers, thin clients, network appliances, and typically embedded systems next to trivial.

LE CHOIX D ’UN LINUX EMBARQUE

• Le choix est à faire en fonction de ses compétences en interne et des TTM à respecter.

• Choisir un linux embarqué commercial est rassurant. Cela a aussi un coût.

LE CHOIX D ’UN LINUX EMBARQUE

Complexité de mise en œuvre maximale

LFS (Linux From Scratch)

µClinux

ELW
Embedded Debian Project, PeeWeeLinux

LRP
Montavista/Professional Edition
Metrowerks/Creation Suite for Linux
LynuxWorks/Bluecat

Complexité de mise en œuvre minimale

PARTIE 3 :
QUAND LE MATERIEL REJOINT LE LOGICIEL

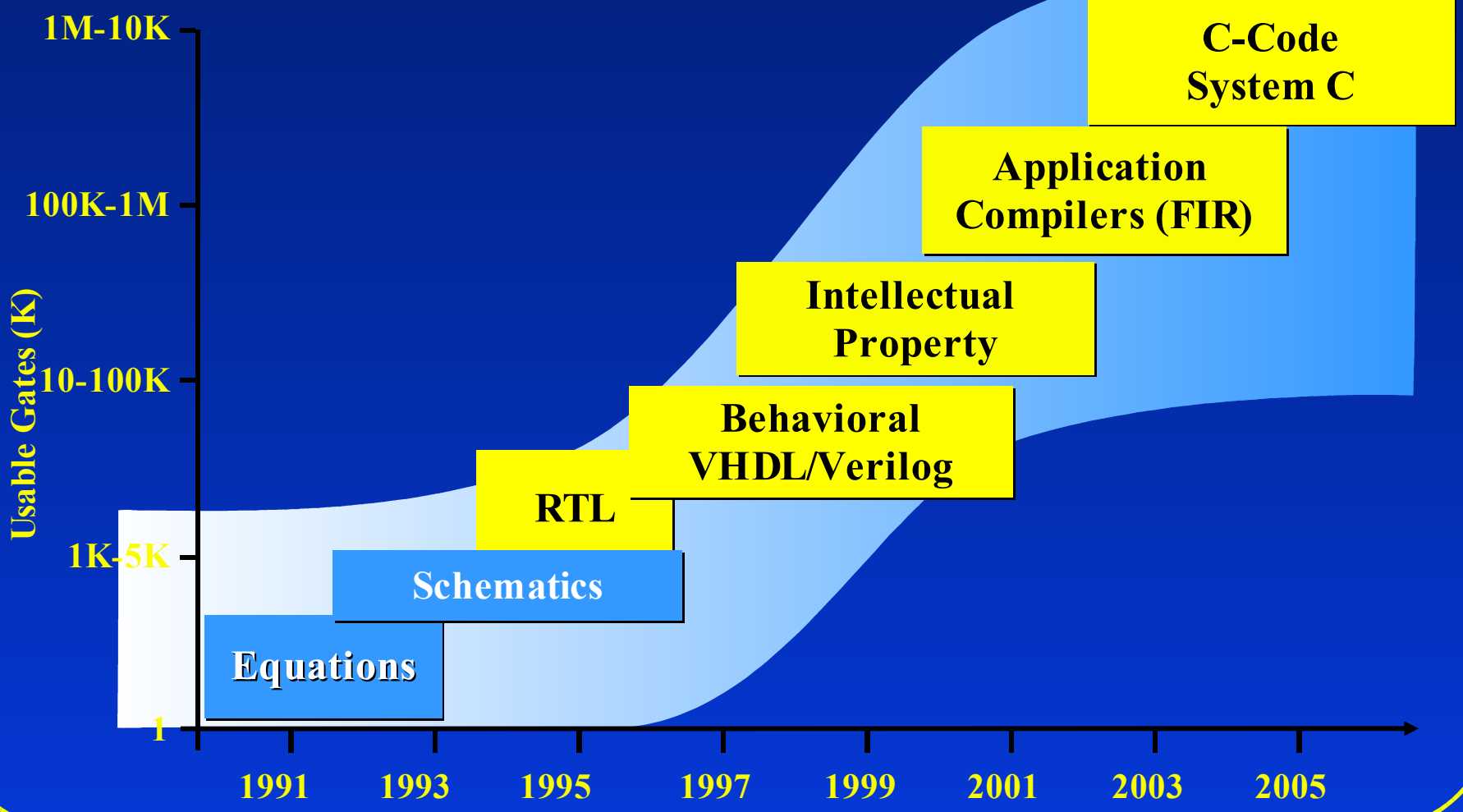
CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL

- La capacité de conception de systèmes numériques permet aujourd'hui de tout intégrer dans un même composant (concept du *single chip*).

- On travaille donc au niveau système et non plus au niveau porte élémentaire ou schématique. On parle de système sur silicium SoC (*System on Chip*) ou SoPC *(System on Programmable Chip*).

- Ceci est lié à la loi empirique de Moore qui dit que pour une surface de silicium donné, on double le nombre de transistors intégrés tous les 18 mois !

CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL



CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL

- On utilise maintenant des langages de description du matériel (VHDL, Verilog) pour synthétiser et aussi tester les circuits numériques. On a ainsi une approche logicielle pour concevoir du matériel.

- Avec l'augmentation de l'intégration, les systèmes numériques se sont complexifiés alors que la mise sur le marché doit être la plus rapide possible :
  - Prise en compte du *Time To Market* (TTM).
  - Réutilisation de choses déjà réalisées (*Design Reuse*).

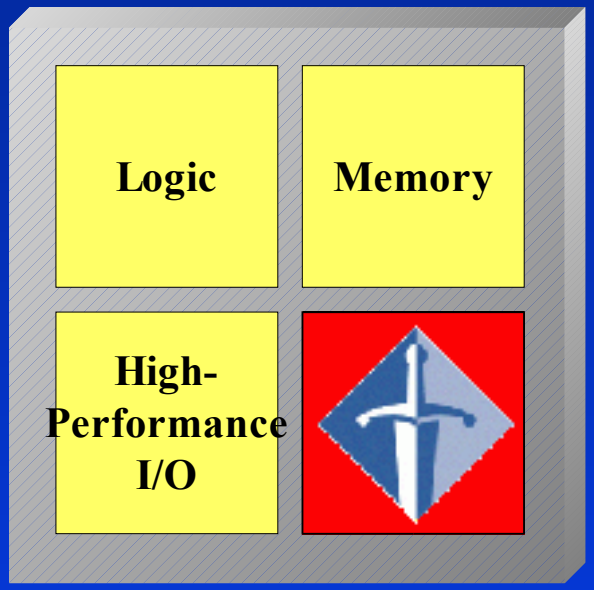
CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL

- On a ainsi vu apparaître la notion de blocs IP (*Intellectual Property*) qui est possible par l'utilisation des langages de description du matériel.

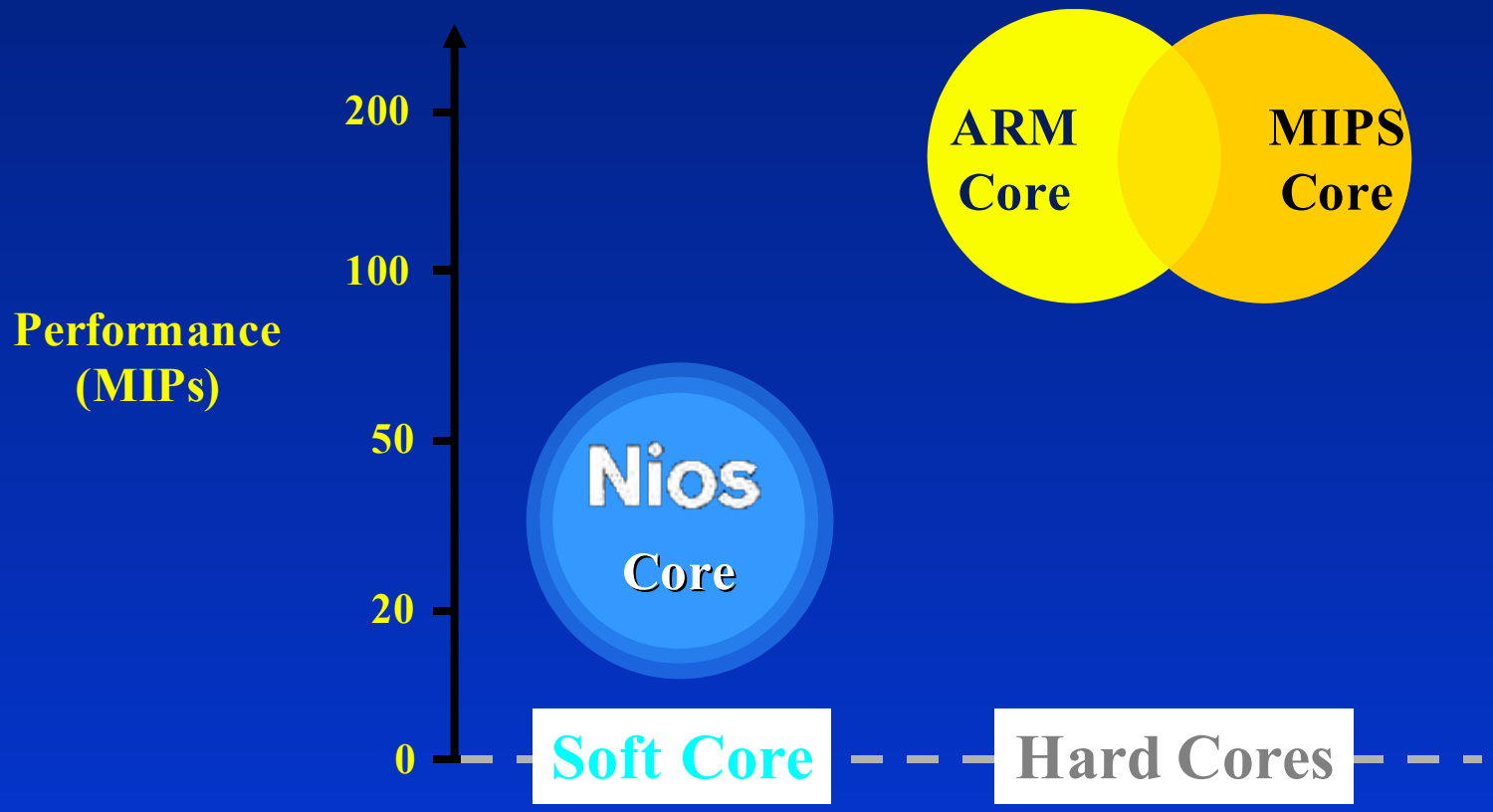
- On achète des blocs IP comme on achète un circuit intégré :
  - interface CAN.
  - DCT.
  - Interface MAC IEEE 802.3 10BaseT qui est la condition nécessaire pour assurer la connectivité IP sur réseau Ethernet.

NIOS D ’ALTERA

- L’offre SoPC Excalibur d’Altera permet la flexibilité de programmation des PLD (*Programmable Logic Device*) avec les performances de temps de traitement d’un processeur embarqué sur silicium pour répondre au besoin d’un court TTM.

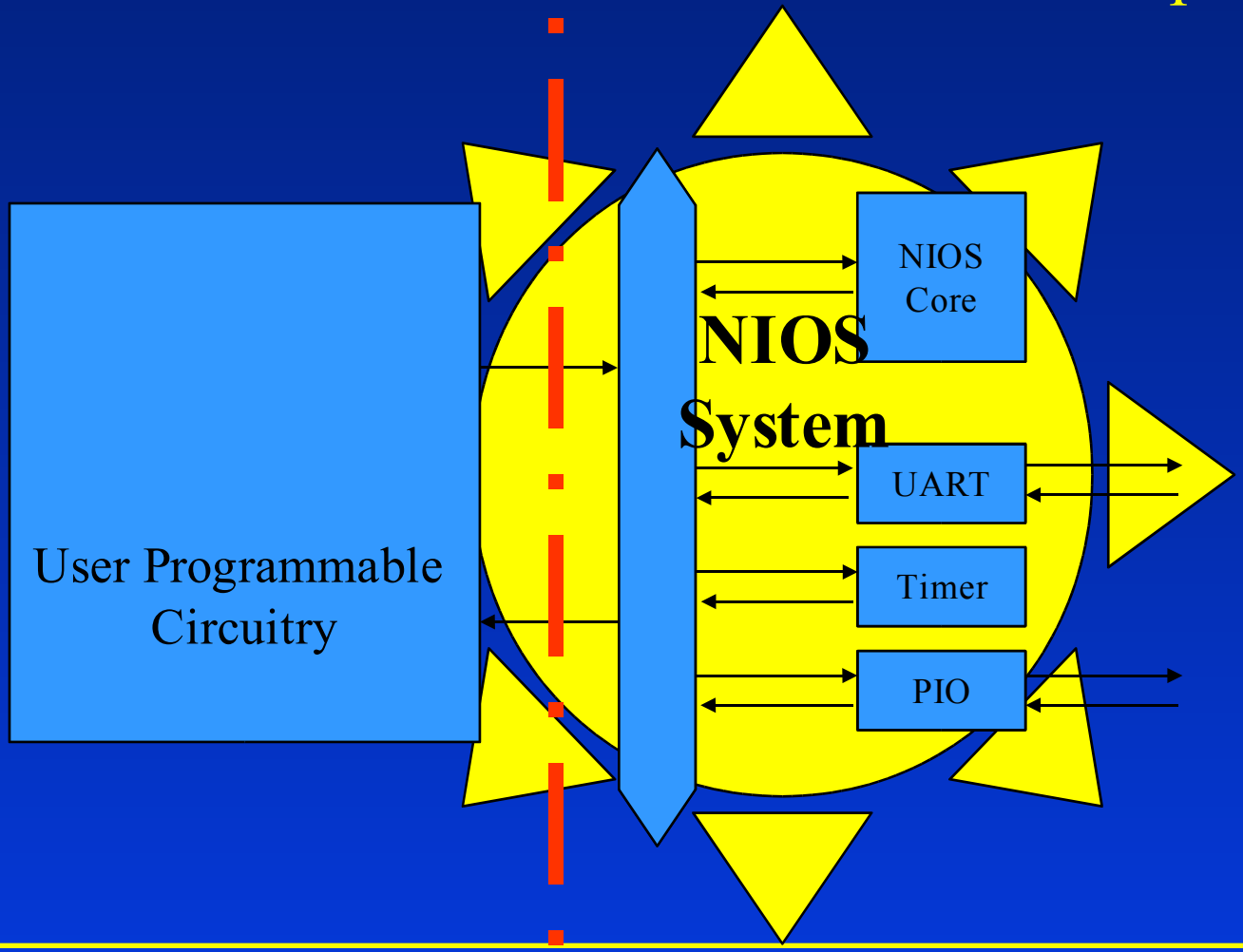


NIOS D 'ALTERA



NIOS D 'ALTERA

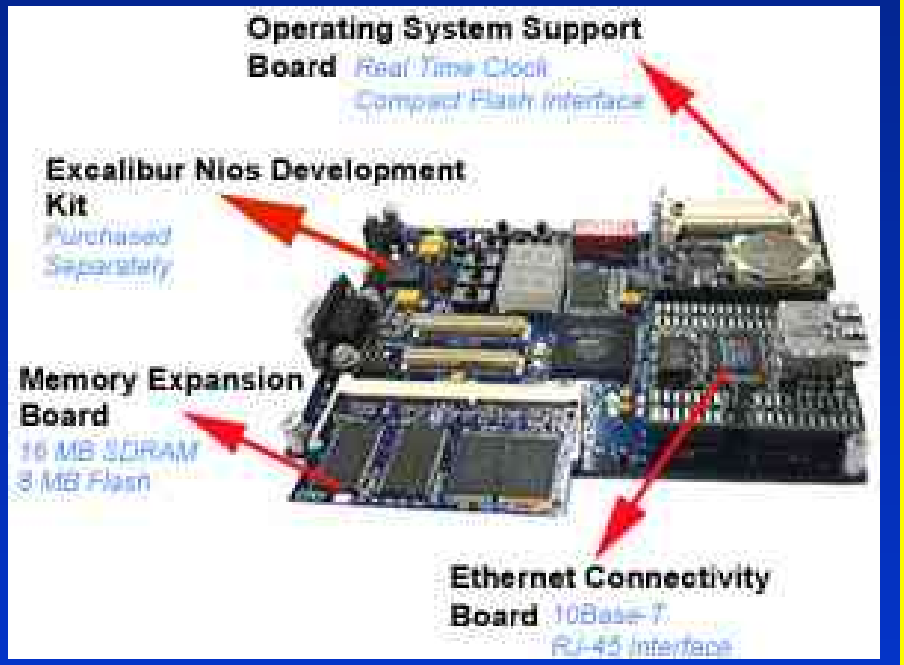
Pour Altéra : « Nios : An Embedded Concept »



NIOS D 'ALTERA

- Linux Development Kit (depuis 09/2001)

• Open-Source µClinux Operating System
• Development Kit Contents
  – µClinux Source Code
  – **Ethernet Development Board**
  – SDRAM / Flash Memory Module
  – SDRAM Controller Core
  – IDE Interface
  – Compact Flash Interface
  – Real Time Clock
  – Reference Design
• Quartus Project
• **Web Server Application**
• Price $2495 (www.microtronix.com)



NIOS D 'ALTERA

- Software Development Tools
  - RedHat GNUPro Toolkit (Compiler, Debugger)
  - Nios Ethernet Development Kit (TCP/IP Stack)

- Operating System Support
  - Linux Development Kit
  - ATI Nucleus
  - µC/OS II

NIOS D ’ALTERA

- L’offre SoPC Excalibur/NIOS d’Altera complétée du portage Linux (µClinux) sur NIOS de Microtronix permet d’avoir une véritable plateforme de Codesign.

- Une interface Ethernet IEEE 802.3 10BaseT (utilisant le composant CS8900A) permet d’avoir naturellement une connectivité IP sous µClinux.

- Il n’existe pas encore un portage GPL de µClinux pour NIOS.

- Il existe par contre un portage µClinux GPL pour le processeur soft Microblaze de Xilinx : http://www.itee.uq.edu.au/~jwilliams/mblaze-uclinux/

CHAPITRE 2 :
LE TEMPS REEL SOUS LINUX

PARTIE 1 :
INTRODUCTION

TEMPS REEL MOU

- Un système d ’exploitation est dit Temps Réel (dur) s ’il est capable de répondre à des sollicitations ou événements (internes ou externes) dans un temps maximum.

- On parle de Temps Réel mou (*Soft Real Time*) quand les événements traités trop tardivement ou perdus sont sans conséquence catastrophique pour la bonne marche du système.

TEMPS REEL MOU

- On peut citer l'exemple des systèmes multimédia : si quelques images ne sont pas affichées, cela ne met pas en péril le fonctionnement correct de l'ensemble du système.

- Dans la très grande majorité des cas, les contraintes de temps sont respectés.

- Ces systèmes se rapprochent fortement des systèmes d'exploitation classiques à temps partagé qui garantissent un temps moyen d'exécution pour chaque tâche (un débit, une Bande Passante). On a ici une répartition **égalitaire** du temps CPU entre processus.

TEMPS REEL DUR

- On parle de Temps Réel dur (*Hard Real Time*) quand les événements traités trop tardivement ou perdus provoquent des conséquences catastrophiques pour la bonne marche du système (perte d ’informations cruciales, plantage…).

- Les systèmes à contraintes *dures* (*hard real time*) ne tolèrent qu’une gestion **stricte et bornée** du temps afin de conserver l'intégrité du service rendu et sont toujours respectés.

- On citera comme exemples les contrôles de processus industriels sensibles comme la régulation des centrales nucléaires ou les systèmes embarqués utilisés dans l'aéronautique.

TEMPS REEL DUR

- Ces systèmes garantissent un temps maximum d'exécution pour chaque tâche.
- On a ici une répartition **totalitaire** du temps CPU entre tâches.
- On peut dire qu'un système temps réel doit être prévisible (*predictible* en anglais*)*, les contraintes temporelles pouvant allant jusqu'à quelques micro-secondes (µs).

LINUX ET LE TEMPS REEL

- Linux standard n ’est pas un système d ’exploitation Temps Réel (dur) car :
  - Le noyau Linux possède de longues sections de code où tous les événements extérieurs sont masqués (non interruptible).
  - Le noyau Linux n ’est pas préemptible durant toute l ’exécution d ’un appel système (structure monolithique) par un processus et ne le redevient qu ’en retour d ’appel système (mode user).

LINUX ET LE TEMPS REEL

- Linux n ’est pas un système d ’exploitation Temps Réel (dur) car :
  - Le noyau Linux n ’est pas préemptible durant le service d ’une interruption (ISR). La routine ISR acquitte l ’interruption puis programme un « Bottom Half » (BH) pour le traitement des données.
  - Le BH programmé par l ’ISR (et éventuellement les autres BH des autres ISR) ne sera exécuté qu ’à la fin de l ’exécution complète de l ’appel système d ’où un temps de latence important et non borné fatal à un système Temps Réel !

LINUX ET LE TEMPS REEL

- Linux n ’est pas un système d ’exploitation Temps Réel (dur) car :
  - L ’ordonnanceur de Linux essaye d ’attribuer de façon équitable le CPU à l ’ensemble des processus (ordonnancement de type *old aging* mise en œuvre pour favoriser l ’accès CPU aux processus récents). C ’est une approche égalitaire. Un ordonnanceur Temps Réel donnera toujours la main à la tâche de plus forte priorité prête. C ’est ici un approche plus totalitaire.

LINUX ET LE TEMPS REEL

- Le noyau Linux standard peut être considéré **par définition** comme Temps Réel (mou) si l ’on travaille avec une réactivité de l ’ordre de la centaine de ms ou plus.

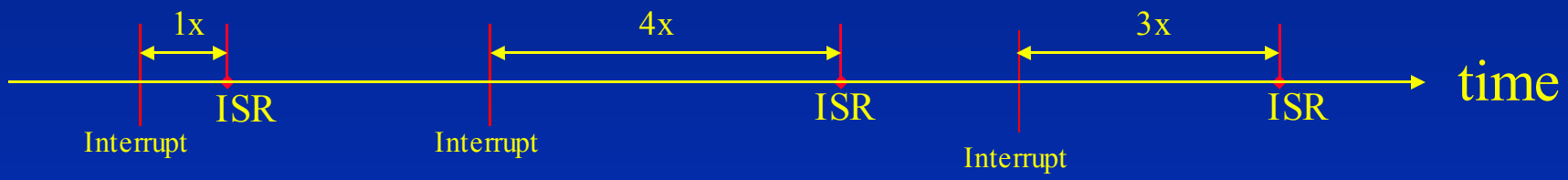
- Il existe des solutions Linux Temps Réel mou par application de patchs dits préemptifs sur un noyau Linux standard pour une réactivité de quelques centaines de µs…

- Il existe des solutions Linux Temps Réel dur pour une réactivité de quelques dizaines de µs…

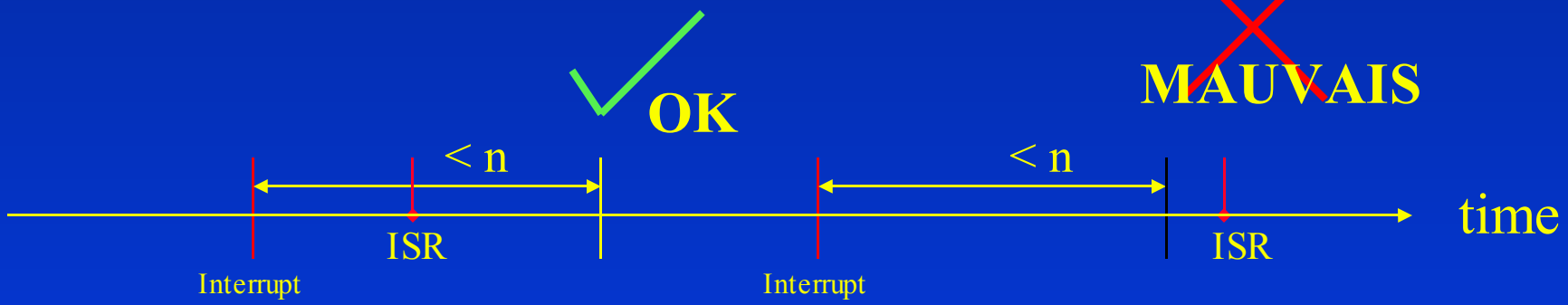
LINUX ET LE TEMPS REEL

- Traitement des interruptions et ISR (*Interrupt Sub Routine*) :

Linux



Système Temps Réel (RTOS)



n: délai maximum

UNE PETITE EXPERIENCE

- Génération d ’un signal périodique sur une broche du port parallèle.
- Le signal généré sur la broche 2 (bit D0) du port parallèle est théoriquement un signal périodique carré de demi-période T/2 de 50 ms.
- On observe à l'oscilloscope le signal suivant sur un système non chargé (AMD Athlon 1500+).

UNE PETITE EXPERIENCE

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <asm/io.h>

#define LPT 0x378

int ioperm();

int main(int argc, char **argv)
{
    setuid(0);
    if (ioperm(LPT, 1, 1) < 0) {
     perror("ioperm()");
     exit(-1);
    }
```

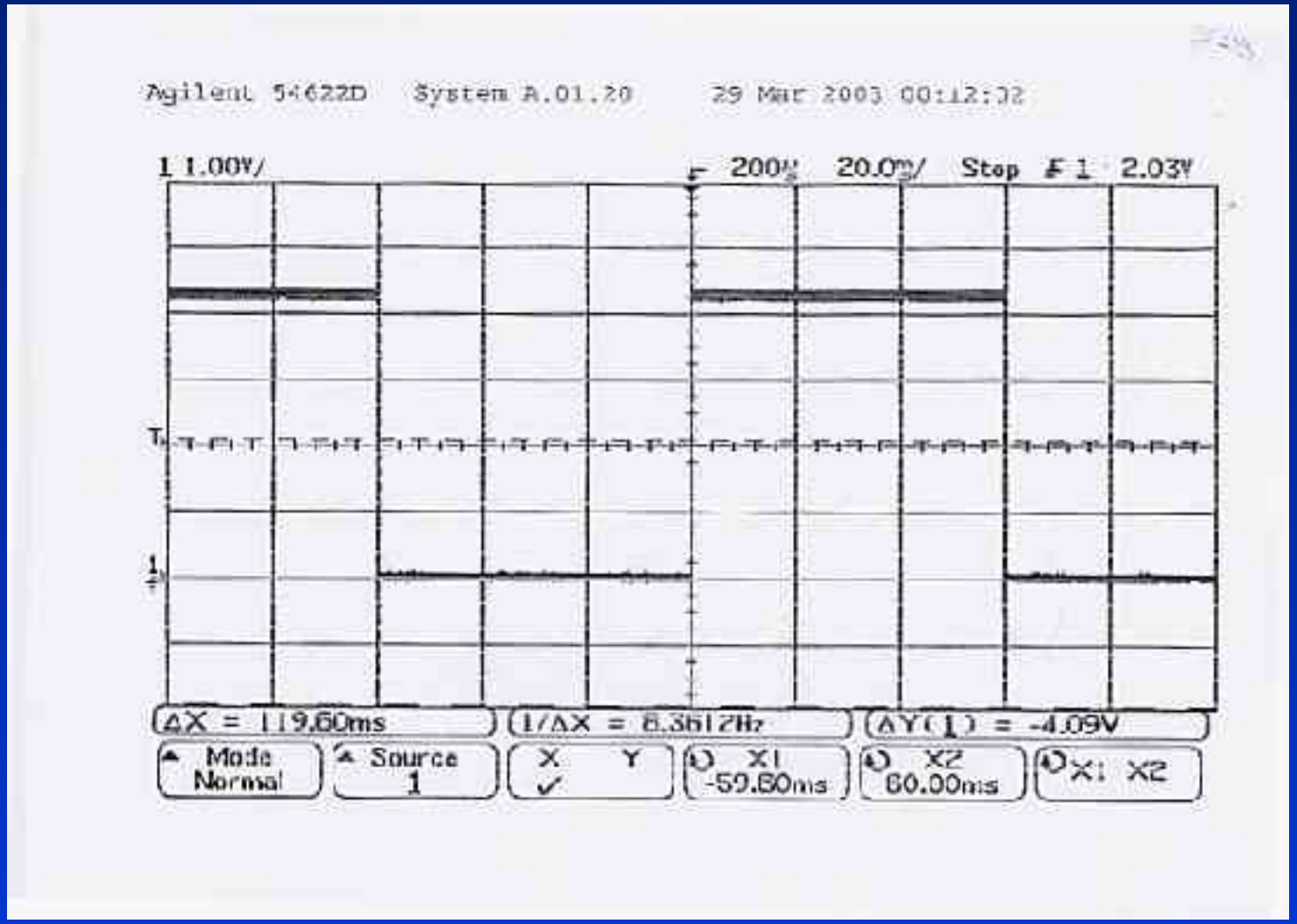
UNE PETITE EXPERIENCE

```
while(1) {
 outb(0x01, LPT);
 usleep(50000);

 outb(0x00, LPT);
 usleep(50000);
}
return(0);
}
```

Programme square (fichier C square.c)

UNE PETITE EXPERIENCE



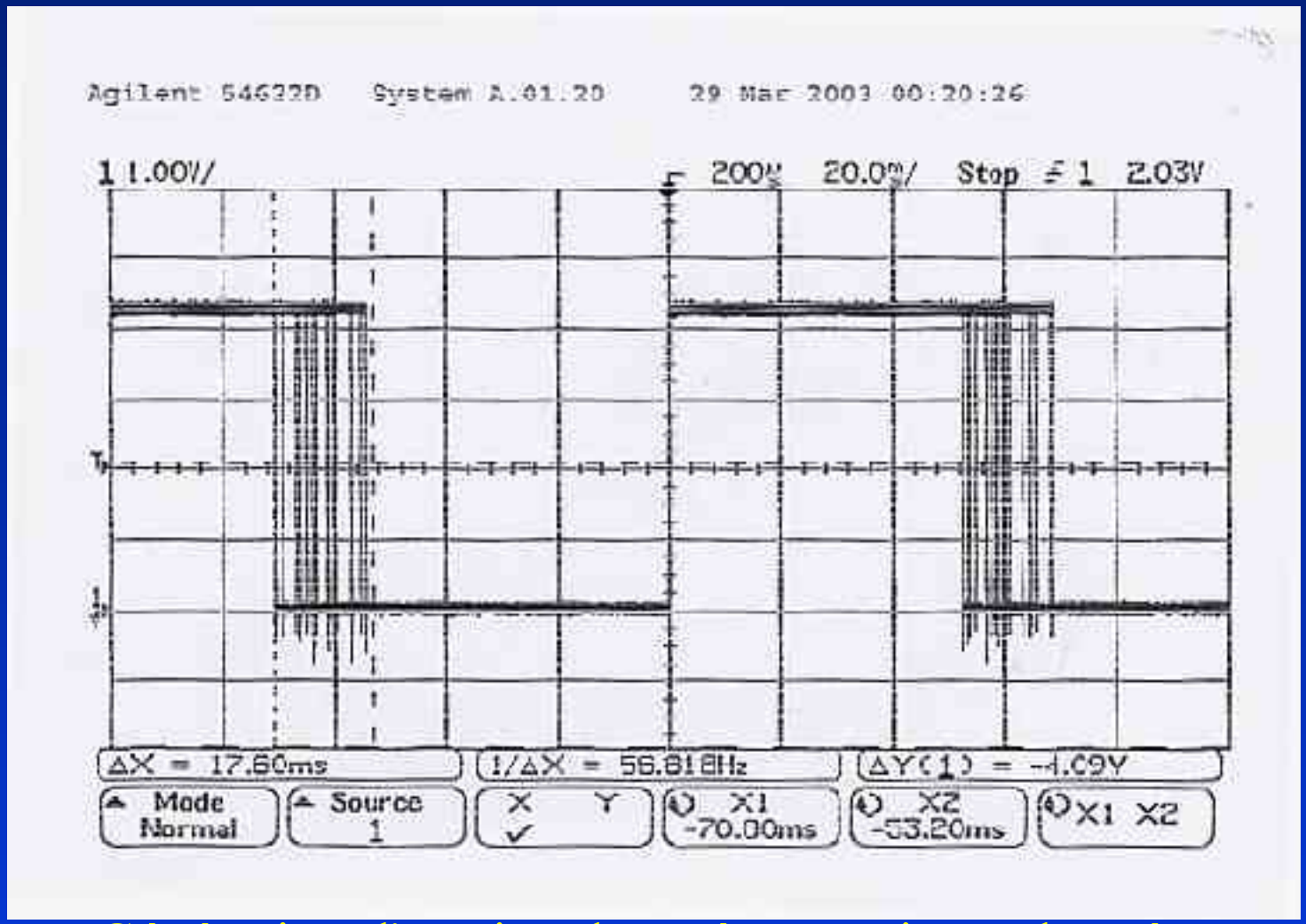
Génération d'un signal carré sous Linux non chargé

UNE PETITE EXPERIENCE

- On remarque que l'on n'a pas une période de 100 ms mais de 119,6 ms dû au temps supplémentaire d'exécution des appels système.

- Dès que l'on stresse le système (écriture répétitive sur disque d'un fichier de 50 Mo), on observe le signal suivant :

UNE PETITE EXPERIENCE



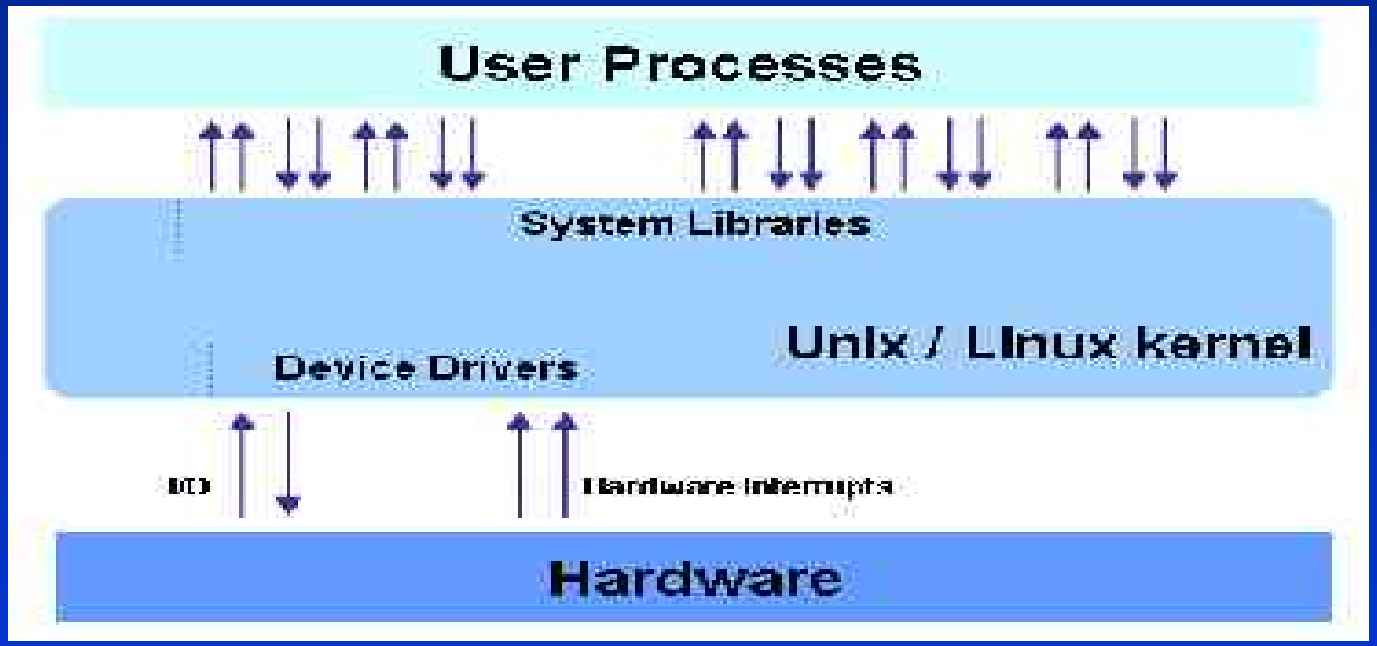
Génération d'un signal carré sous Linux chargé

UNE PETITE EXPERIENCE

- On observe maintenant une gigue (*jitter*) sur le signal généré. La gigue maximale sur la durée de l'expérience est de 17,6 ms.
- La forme du signal varie maintenant au cours du temps, n'est pas de forme carrée mais rectangulaire. Linux n'est donc plus capable de générer correctement ce signal.
- Il faut noter aussi que le front montant sur la figure précédente apparaît sans gigue car il a servi comme front de synchronisation de l'oscilloscope. La gigue observée est donc à voir comme la contribution de la gigue sur front montant et sur front descendant.
- Si l'on diminue la valeur de la demi-période, la gigue devient aussi importante que cette dernière et dans ce cas, Linux ne génère plus aucun signal !

EXTENSION TEMPS REEL POUR LINUX

- Implémentation du noyau Linux standard :
  - Pas de support du Temps Réel.
  - Séparation entre le matériel et les processus Linux.
  - ...



Linux standard

EXTENSION TEMPS REEL POUR LINUX

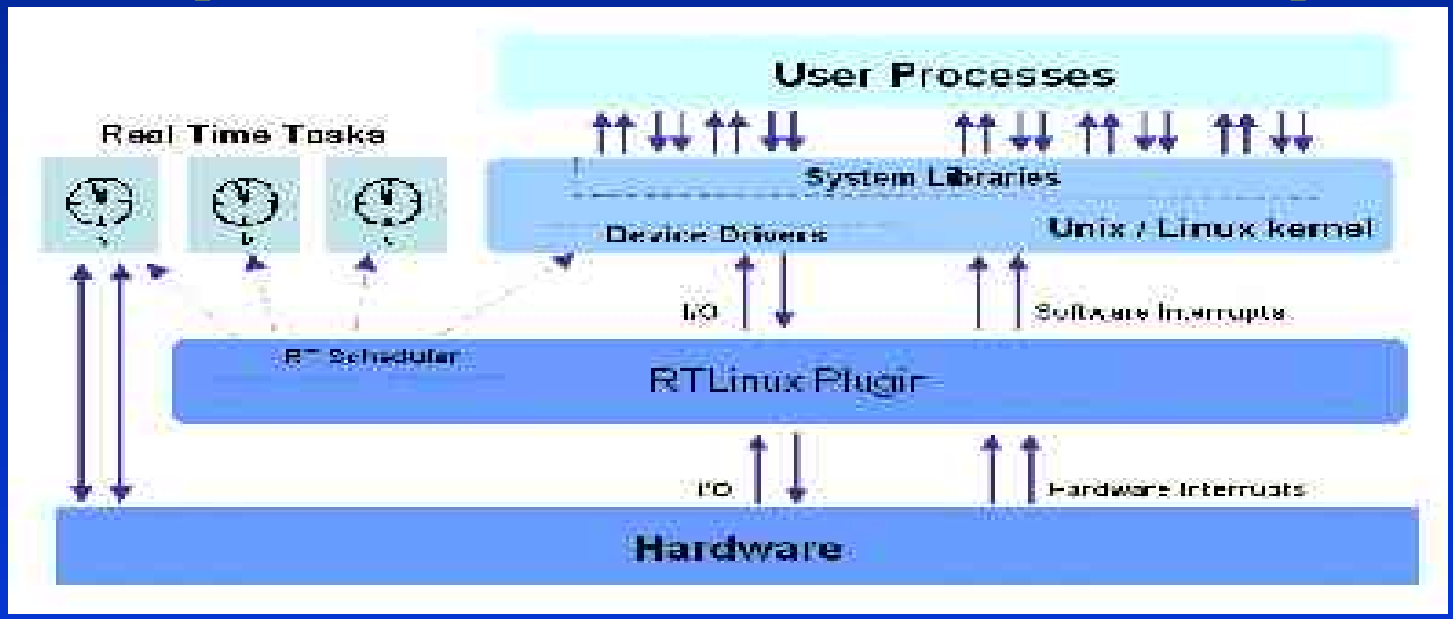
- Solution 1 pour une extension Temps Réel mou de Linux :
  - Modification du noyau Linux par application de patchs pour améliorer les performances Temps Réel : dévalider les interruptions le moins longtemps possible, appeler l 'ordonnanceur le plus souvent possible (fonction *schedule()* du noyau), en retour d 'interruption par exemple.
  - Les patchs dits préemptifs permettant d'améliorer le comportement du noyau Linux en réduisant les temps de latence de ce dernier. Ces modifications ne transforment pas Linux en noyau temps réel dur mais permettent d'obtenir des résultats satisfaisants dans le cas de contraintes temps réel molles (respect des contraintes de temps dans la très grande majorité des cas).

EXTENSION TEMPS REEL POUR LINUX

- Solution 2 pour une extension Temps Réel dur de Linux :
  - Ajout d ’un deuxième ordonnanceur TR de tâches et considérer le noyau Linux et ses processus comme tâche de fond. Plus difficile que la première solution.
  - Cette technique permet de mettre en place des systèmes temps réel durs.
  - Utilisé dans les projets RTLinux et RTAI par exemple.
  - On ne peut pas considérer Linux et son extension TR dans ce cas comme un véritable Noyau TR monolithique (pour les puristes du TR) et enfreint la « logique Linux » et la cohérence de l ’API Linux (pour les puristes Linux)...

EXTENSION TEMPS REEL POUR LINUX

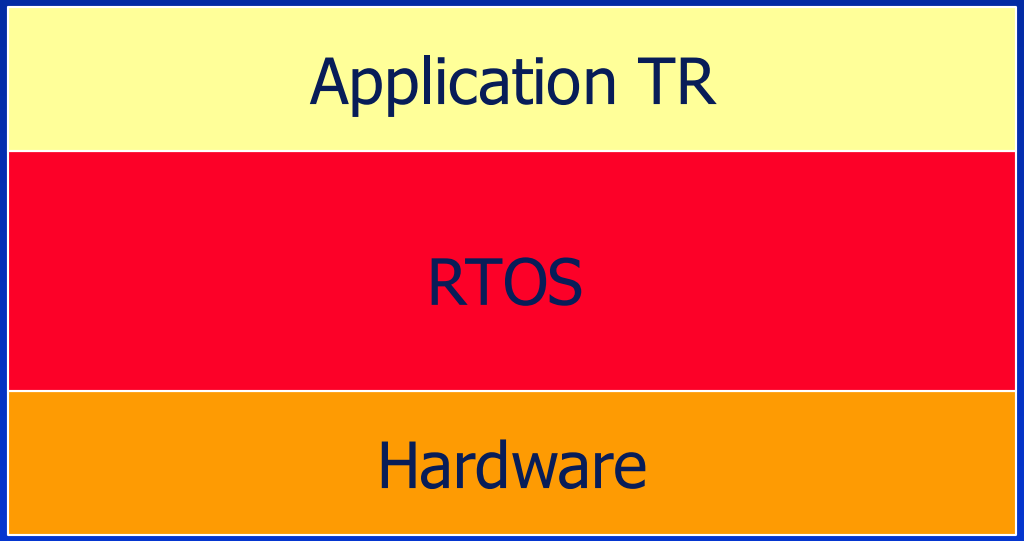
- Solution 2 pour une extension Temps Réel dur de Linux :
  - Ajout d ’une couche d ’abstraction entre le matériel et le noyau Linux.
  - Définition de tâches Temps Réel.
  - Pas de séparation entre le matériel et les tâches Temps Réel.



Linux et un extension TR de type RTLinux

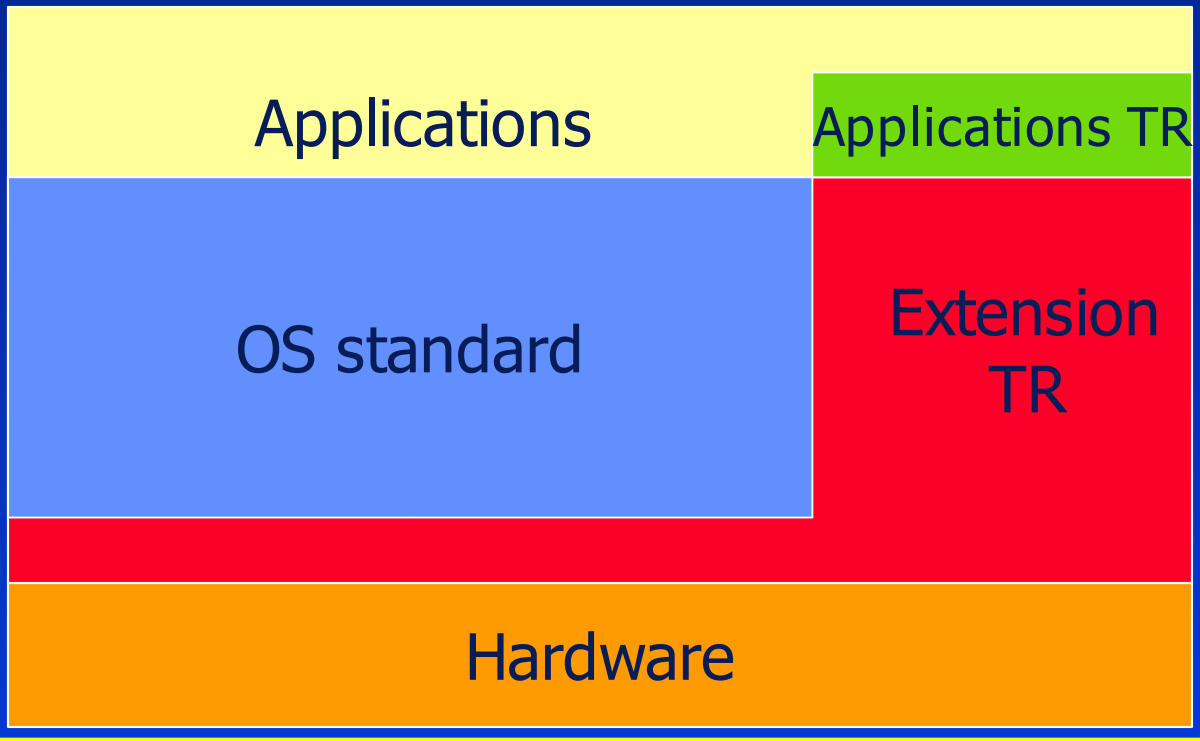
VERITABLE RTOS

- Avantages : simplicité, monolithique, fait pour le TR, petit overhead.
- Inconvénients : fonctionnalités limitées.
- Exemples : VxWorks, QNX, pSOS, VRTX, µC/OS II...



RTOS PAR EXTENSION

- Avantages : de nombreuses fonctionnalités, coopération entre tâches TR et processus non TR.
- Inconvénients : n ’est pas un vrai RTOS monolithique.
- Exemples : RTLinux, RTAI...



PARTIE 2 :
LES PATCHS PREEMPTIFS

PATCH DU NOYAU

- Il existe deux principaux patchs permettant d'améliorer la réactivité du noyau Linux :
  - Patch Preempt Kernel
  - Patch Low Latency

PATCH DU NOYAU

- Le patch Preempt Kernel est maintenu par Robert M. Love et soutenu par MontaVista :

  http://www.tech9.net/rml/linux

- Le principe du patch est de rendre le noyau totalement préemptible et de protéger les données du noyau par des mutexs (ou spinlocks).

- A chaque fois qu ’un événement apparaît et rend un processus de plus forte priorité prêt, le noyau préempte le processus courant et exécute le processus de plus forte priorité.

PATCH DU NOYAU

- Le patch Low Latency est maintenu par Andrew Morton : http://www.zip.com.au/~akpm/linux/schedlat.html

- Le principe est un peu différent car au lieu d'opter pour une stratégie systématique du noyau tout préemptif, les développeurs du patch ont préféré effectuer une analyse du code source du noyau afin d'ajouter des points de préemption obligatoire (appel de schedule()) subtilement placés dans les sources du noyau afin de casser des boucles non préemptibles trop longues.

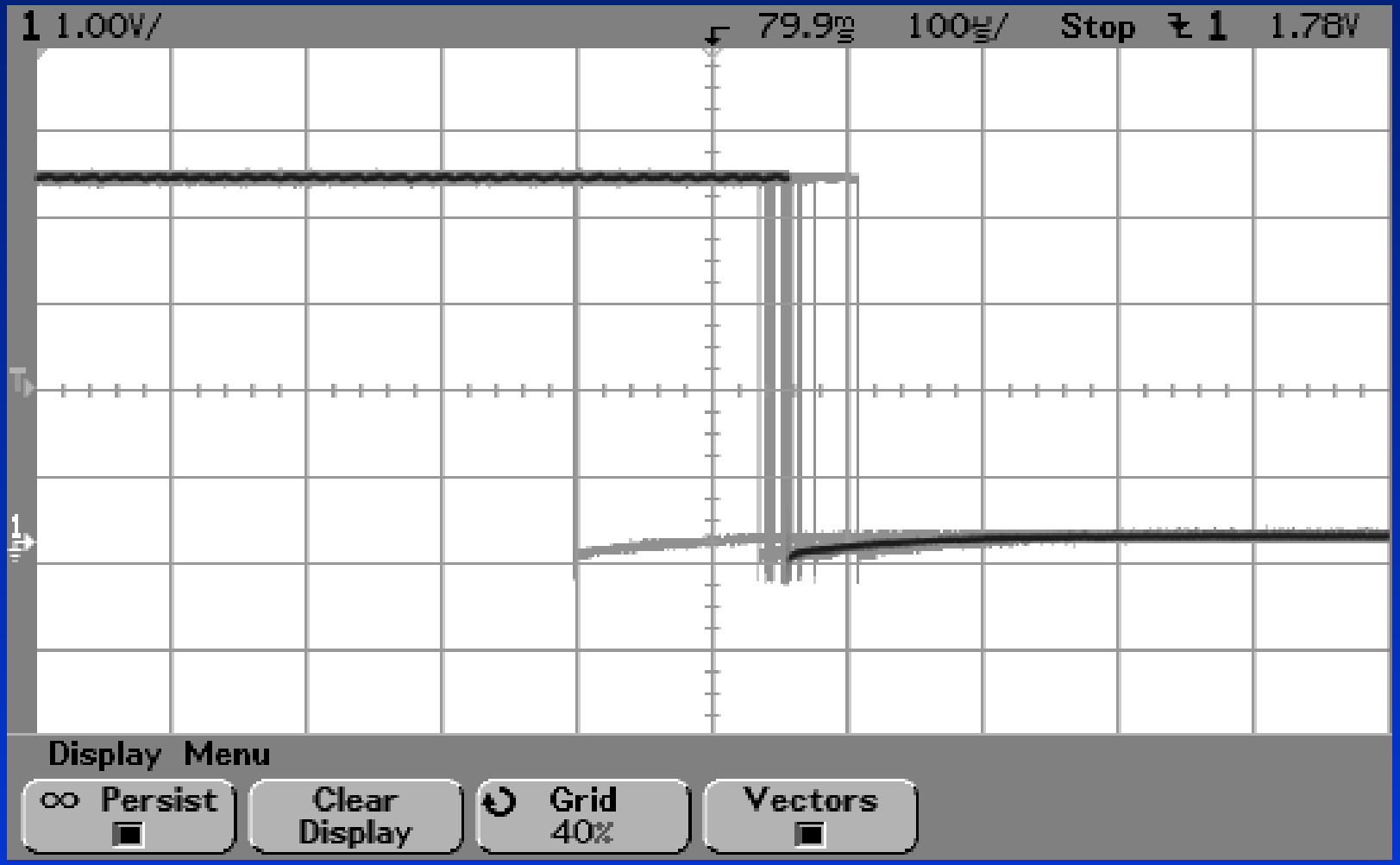
PETITE EXPERIENCE. SUITE

- Mesure effectuée à l'oscilloscope lors de l'utilisation du programme square décrit précédemment.

- Dans le cas du noyau 2.4.20 modifié par le patch *Preempt Kernel* et subissant la même charge que pour les autres mesures, nous obtenons la courbe suivante, indiquant une latence maximale légèrement supérieure à 200 µs.

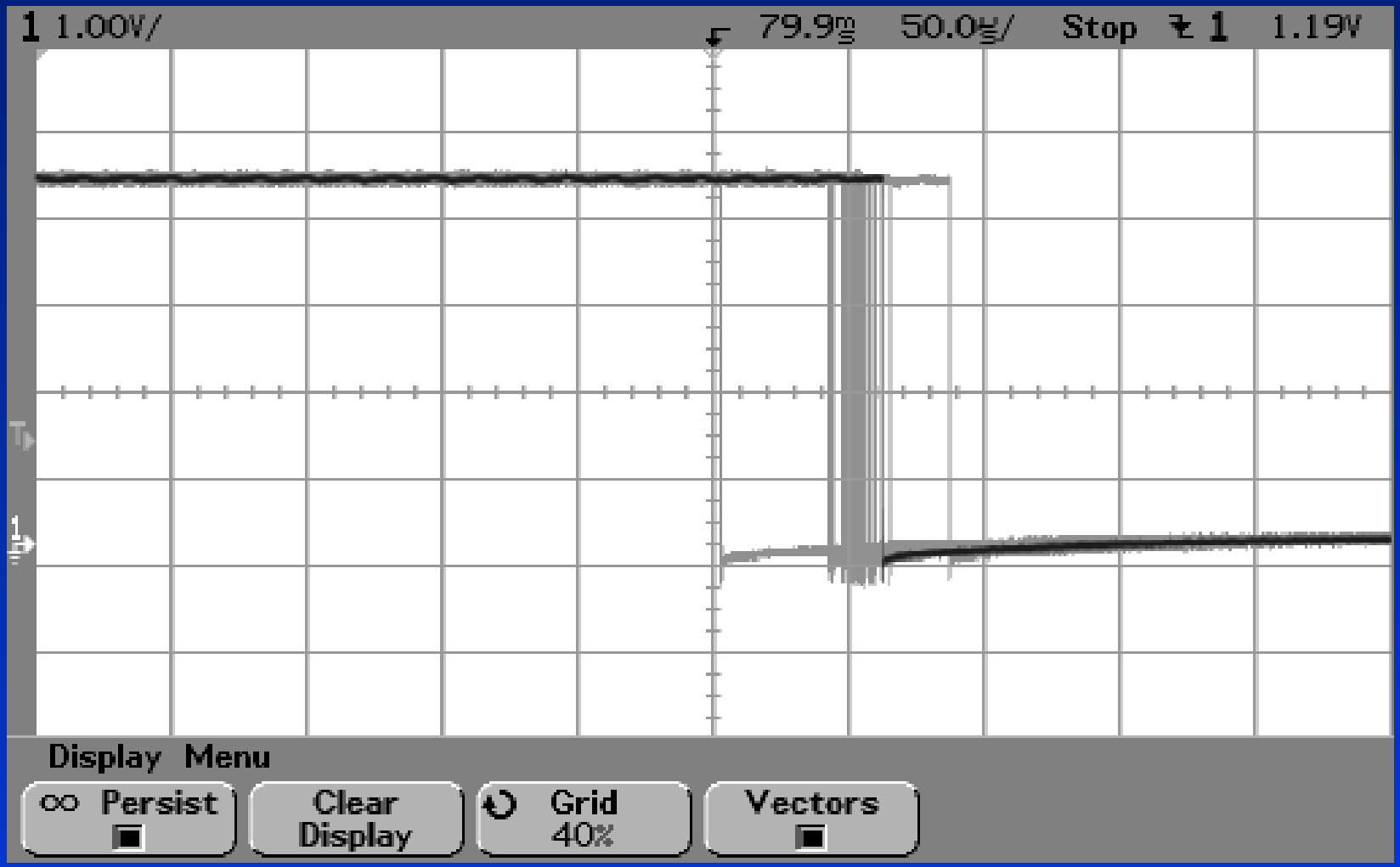
- Dans le cas du patch L*ow Latency*, nous obtenons un meilleur résultat avec une latence maximale d'environ 80 µs.

PETITE EXPERIENCE. SUITE



Exécution de square avec le patch Preempt Kernel

PETITE EXPERIENCE. SUITE



Exécution de square avec le patch Low Latency

CONCLUSION

- Les patchs précédents permettent d'améliorer les temps de latence sur le noyau Linux standard mais le concept se rapproche plus d'une amélioration de la qualité de service que du temps réel dur.
- La réactivité est maintenant de l ’ordre de quelques dizaines à quelques centaines de µs au lieu de quelques dizaines à quelques centaines de ms voire plus pour un noyau Linux standard !

- Des mesures faites par Metrowerks à l ’aide du Latency Benchmark de Systems Software Labs montrent que dans 99,5 % des cas, le temps de latence est inférieur à 200 µs pour les 2 patchs (voir le whitepaper *Linux as a Real-Time operating System* de Metrowerks).
- **Une solution Temps Réel dur donnera un temps de latence inférieur à x µs dans 100 % des cas !**

CONCLUSION

- La mise en œuvre d ’un patch préemptif est à voir comme une solution intermédiaire où la mise en œuvre et la programmation restent simples pour développer des applications TR.

- Une fusion des deux patchs est intégrée dans le futur noyau 2.6.

PARTIE 3 :
LES OFFRES LINUX TEMPS REEL

LES OFFRES LINUX TEMPS REEL

- Les offres de version de Linux embarqué et Temps Réel peuvent être rangées dans l'une des 2 catégories suivantes :
  - Les distributions Linux Temps Réel commerciales.
  - Les distributions Linux Temps Réel libres.

LINUX TEMPS REEL COMMERCIAL

- Montavista/ Professional or Carrier Grade or Consumer Electronics Edition (ex Hard Hat Linux)
- Lineo-Metrowerks-Motorola/Creation Suite for Linux (ex Embeddix)
- LynuxWorks/BlueCat RT
- TimeSys/Linux RTOS Professional or Standard Edition
- RTLinux/Pro

- Il y a toujours les solutions TR commerciales non Linux
  - pSOS/VxWorks, QNX, LynxOS...

LINUX TEMPS REEL OPEN SOURCE

- FSMLabs/RTLinux/free (ex OpenRTLinux GPL)
  http://fsmlabs.com/community/
- RTAI : Real Time Application Interface
  http://www.aero.polimi.it/~rtai/

- eCOS
  http://sources.redhat.com/ecos/
- KURT. Kansas University Real-Time Linux
  http://www.ittc.ku.edu/kurt/
- ADEOS
  http://www.nongnu.org/adeos/
  The purpose of Adeos is to provide a flexible environment for sharing hardware resources among multiple operating systems, or among multiple instances of a single OS.

TEMPS REEL COMMERCIAL : VxWorks ET pSOS

- Solution commerciale TR non Linux. Noyau TR. WindRiver Systems.
  - http://www.wrs.com
- L'encore Numéro 1 dans le domaine du TR et de l'embarqué…

- VxWorks :
  - Scalable (simple to complex designs)
  - Reliable (mission-critical applications, ABS)
  - CPU's PowerPc, 68K, CPU32, ColdFire, MCORE, 80x86 and Pentium, i960, ARM and StrongARM, MIPS, SH, SPARC, NECV8xx, M32 R/D, RAD6000, ST 20, TriCore)
  - Graphic Development Platform
  - Cross-development
  - Support/Documentation
  - POSIX 1003.1b compliant
  - Networking
  - Tornado II embedded development platform

TEMPS REEL COMMERCIAL : QNX

- Solution commerciale TR non Linux. Système TR. QNX software Systems.
  - http://www.qnx.com

- QNX :
  - Highly reliable
  - all generic x86 based processors(386+)
  - Scalable (modules)
  - Deterministic
  - "QNX Neutrino™ real-time OS, "the most advanced RTOS on the market"
  - Networking
  - Graphical development tools and debugger
  - Visual design tools (C code form cut and paste)

TEMPS REEL COMMERCIAL : LynxOS

- Solution commerciale TR compatible Linux. Système TR. LynuxWorks Systems (ex Lynx).
  - http://www.lynxos.com/

- LynxOS is unique in the real-time embedded software marketplace. It is a hard RTOS that combines performance, reliability, openness, and scalability together with patented technology for real-time event handling. Flexible scalability makes the LynxOS well suited for applications ranging from large and complex switching systems down to small highly embedded products. LynxOS is binary compatible with the BlueCat Linux, enabling users to take advantage of the best configuration for their needs. In addition, LynuxWorks also supports traditional UNIX and Java and supports processors from Intel, Motorola, and MIPS. LynxOS offers users a choice of software application interfaces, a large number of development tools, scalability and memory efficiency which reflect the many years of expertise LynuxWorks has in the real-time embedded systems market.

LINUX TEMPS REEL COMMERCIAL

- MontaVista/Professional or Carrier Grade or Consumer Electronics Edition :
  - Solution générale (et TR) pour l'embarqué
  - http://www.mvista.com/
  - kit d 'évaluation disponible (preview kit)

- MontaVista Linux Professional Edition
- MontaVista Linux Carrier Grade Edition
- MontaVista Linux Consumer Electronics Edition

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques TR de MontaVista/Professional Edition :
- Solution 1 : application du patch Preempt Kernel
  - The MontaVista Linux preemptible kernel is available and shipping for all supported architectures: PowerPC, x86, MIPS, StrongARM, XScale, SH and ARM

- Plus d ’infos : http://www.linuxdevices.com/articles/AT4185744181.html
- Le patch kpreempt de Montavista est sous licence GPL :
- http://www.tech9.net/rml/linux/

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques TR de MontaVista/Professional Edition :
- Solution 2 : redéfinition des politiques d'ordonnancement (fixed overhead scheduler)
  - Scheduler TR qui gère tous les threads Linux marqués SCHED_FIFO et SCHED_RR basé sur leur priorité fixes (128). Les threads marqués différemment sont traités par le scheduler Linux standard.
  - On n 'étend pas l 'API Linux standard.
  - Emulation de l 'API VxWorks et pSOS.
- Le scheduler RT de Montavista est sous licence GPL :

http://sourceforge.net/projects/rtsched

LINUX TEMPS REEL COMMERCIAL

- Lineo-Metrowerks-Motorola/Creation Suite for Linux :
  - http://www.metrowerks.com/
  - kit d ’évaluation disponible

- L ’outil de configuration du noyau LKIT (Linux Kernel Import Tool) supporte l ’application de patchs du noyau donc les patchs préemptifs.

LINUX TEMPS REEL COMMERCIAL

- LynuxWorks/BlueCat RT:
  - Extension Temps Réel de BlueCat
  - http://www.bluecat.com/products/bluecat-rt/bluecat-rt.php3

- Mise en œuvre de l'extension TR par double noyau.
- Utilisation d'un noyau TR (licence RTLinux/Pro).

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques de LynuxWorks/BlueCat RT :

LINUX TEMPS REEL COMMERCIAL

- TimeSys/Linux RTOS Professional ou Standard Edition :
  - http://www.timesys.com

- Les 2 produits sont basés sur un noyau Linux modifié pour le Temps Réel :
  - TimeSys Linux GPL

- Application d'un patch propriétaire de type kpreempt pour rendre le noyau préemptif : Temps Réel mou.

RTLinux COMMERCIAL ET OPEN SOURCE

- RTlinux/Free Pro :
- Solution libre et commerciale d'extension TR de Linux. Système TR. FSM Labs
- http://www.rtlinux.org
- Mise en place d 'une couche d 'abstraction.
- Mise en service sous forme de modules Linux.
- Linux apparaît comme la tâche de fond de plus faible priorité.
- RTLinux propose une API cohérente pour une programmation TR.
- Existe en une version tenant sur une disquette : projet miniRTL.

RTAI

- Solution libre d'extension TR de Linux. Système TR. Université de Milan en Italie.
- http://www.aero.polimi.it/~rtai/
- Mise en place d 'une couche d 'abstraction
- Mise en service sous forme de modules Linux :
- Linux apparaît comme la tâche de fond de plus faible priorité.

PARTIE 4 :
PRESENTATION DETAILLEE DE RTLINUX

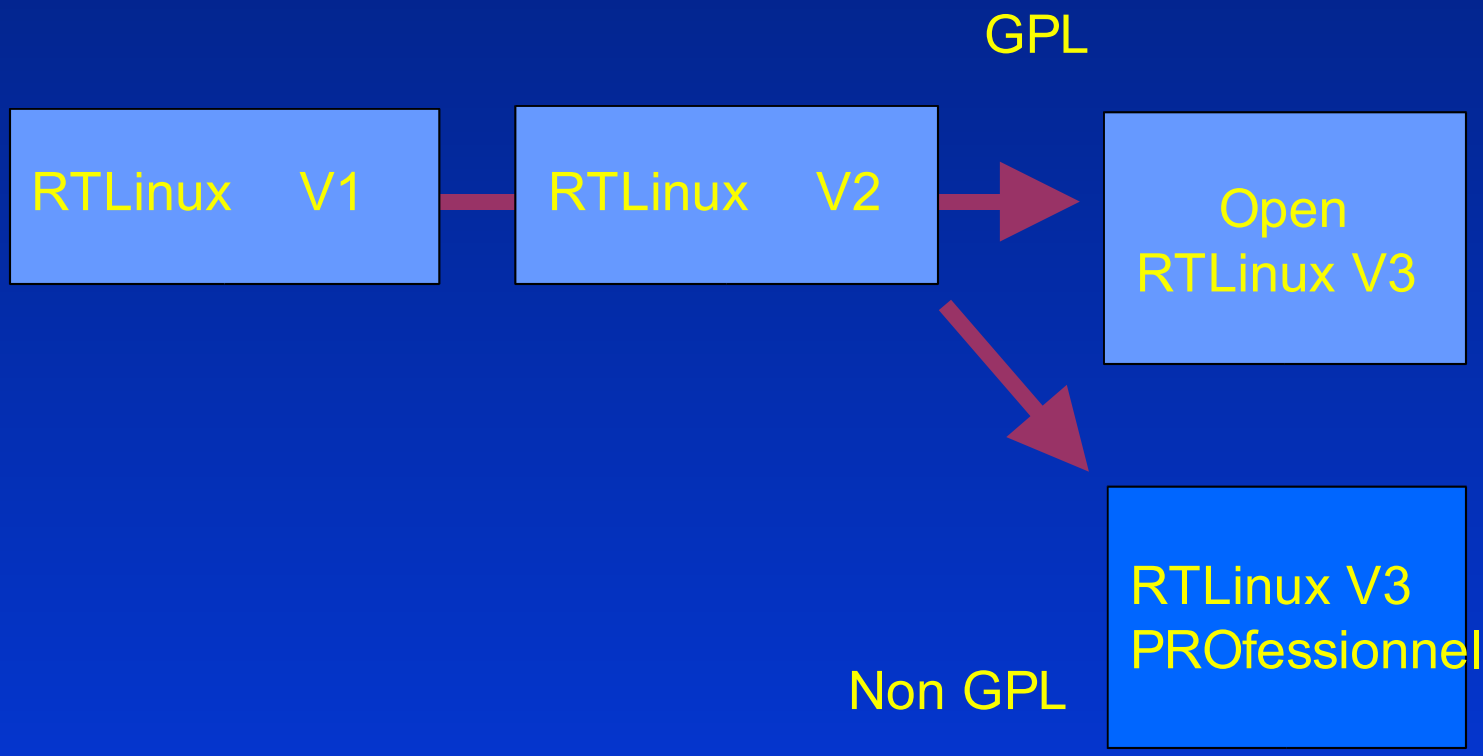
HISTOIRE DE RTLinux

- RTLinux a été développé originellement par un chercheur de l'université de New Mexico avec l'aide d'un étudiant : Victor Yodaiken et Michael Barabanov.

- D 'abord sous licence GPL, un brevet (US Patent No. 5,995,745) a été déposé sur le principe de fonctionnement de RTLinux, ce qui est incompatible avec la notion de logiciel libre. Une entreprise privée FSMLabs a été créée pour distribuer RTLinux.

- Devant le tollé général, FSMLabs décide de distribuer une version GPL OpenRTLinux et une version commerciale RTLinux/PRO plus complète.

HISTOIRE DE RTLinux



HISTOIRE DE RTLinux

- RTLinux V 1 :
  - Noyau Linux 2.0.x.
  - Pas de support SMP (*Symmetric Multi Processor*).
  - API simple de près 15 fonctions.
- RTLinux V2 :
  - Noyau Linux 2.2.x.
  - SMP.
  - API style POSIX.
- RTLinux V3 :
  - Noyau Linux 2.2.19 &2.4.4 (26/11/2001).
  - Version GPL et PRO.

FONCTIONNALITES DE RTLinux

- Mise en service sous forme de modules Linux.
- Les tâches TR sont chargées comme des modules Linux.
- Linux apparaît comme la tâche de fond.
- RTLinux propose une API simple pour une programmation TR. L'API POSIX thread est supportée pour l'écriture de tâches TR.
- Les communications entre processus Linux et les tâches TR se font par des FIFOS.
- Le dernière version stable supporte les processeurs x86, PowerPC et Alpha.

IPC RTLinux

- Les Communications Inter Processus IPC se font par des FIFOS RT.
- Les buffers des FIFOS RT sont alloués dans l'espace noyau.
- Le nombre maximal de FIFOS est fixe (fixé à la compilation du noyau).
- Une FIFO est unidirectionnelle. Pour des communications bidirectionnelles, il en faut donc 2.

IPC RTLinux

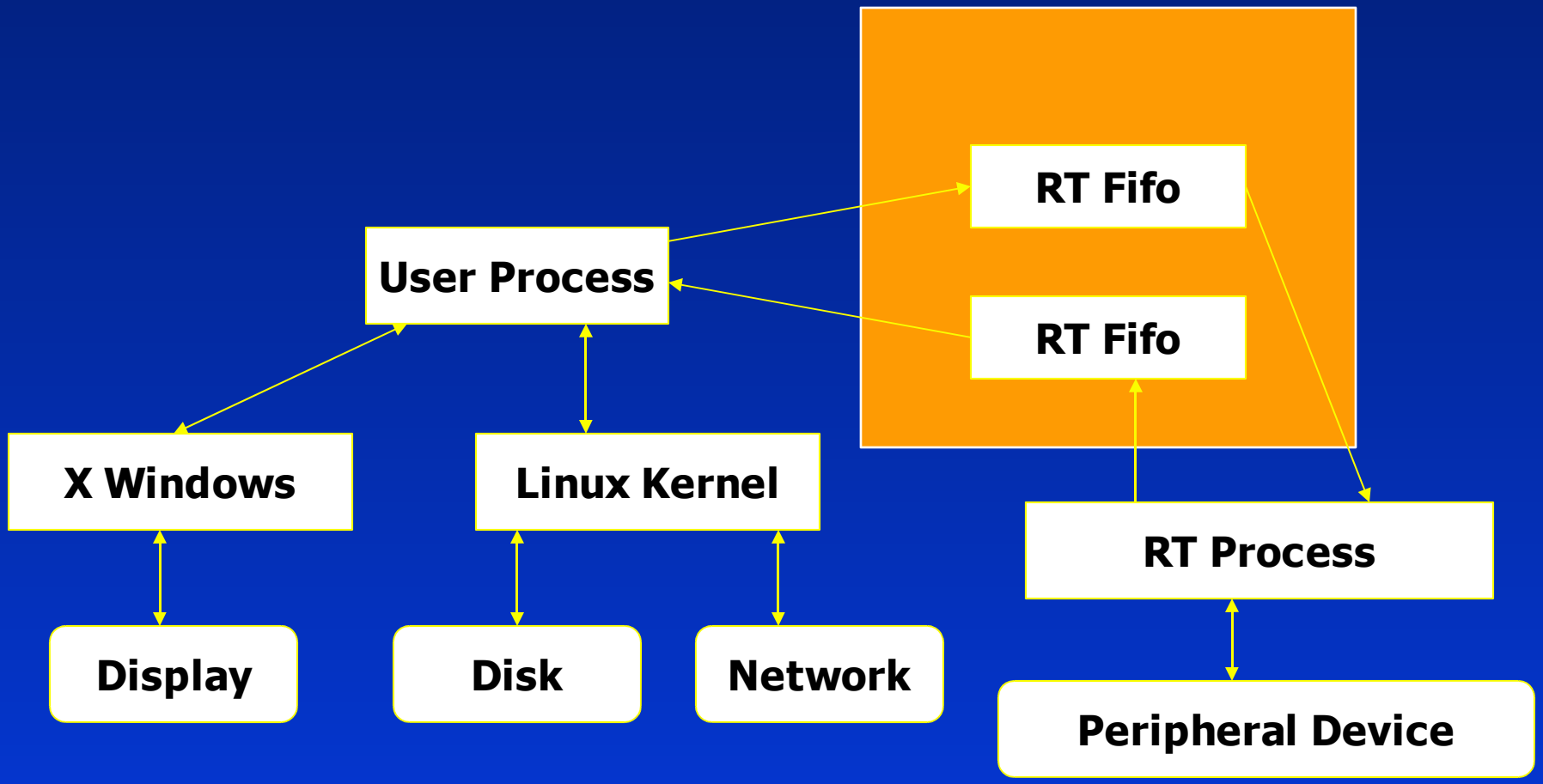
- Des IPC par mémoire partagée sont possibles entre processus Linux et tâches RTLinux.
- Il convient de charger le module mbuff et de travailler avec le périphérique /dev/mbuff.
- Un processus Linux peut mapper une zone mémoire allouée dans l'espace noyau dans son propre espace d'adressage.
- La mémoire allouée dans l'espace noyau n'est pas forcément physiquement contigue. Elle ne peut être swappée.

DEVELOPPEMENT SOUS RTLinux

- Le développement d'applications sous RTLinux suit les règles suivantes :
  - Découpage en 2 parties : TR et non TR.
  - La partie TR doit être la plus simple et la plus courte possible.
  - Le reste est non TR et sera développé dans l'espace user sous forme de processus Linux.
  - Les processus Linux et les tâches TR pourront communiquer par des FIFOS ou par mémoire partagée.

- L'API POSIX thread existant déjà sous Linux a été portée sous RTLinux. Cela facilite la migration d'un développeur Linux vers RTLinux.

DEVELOPPEMENT SOUS RTLinux

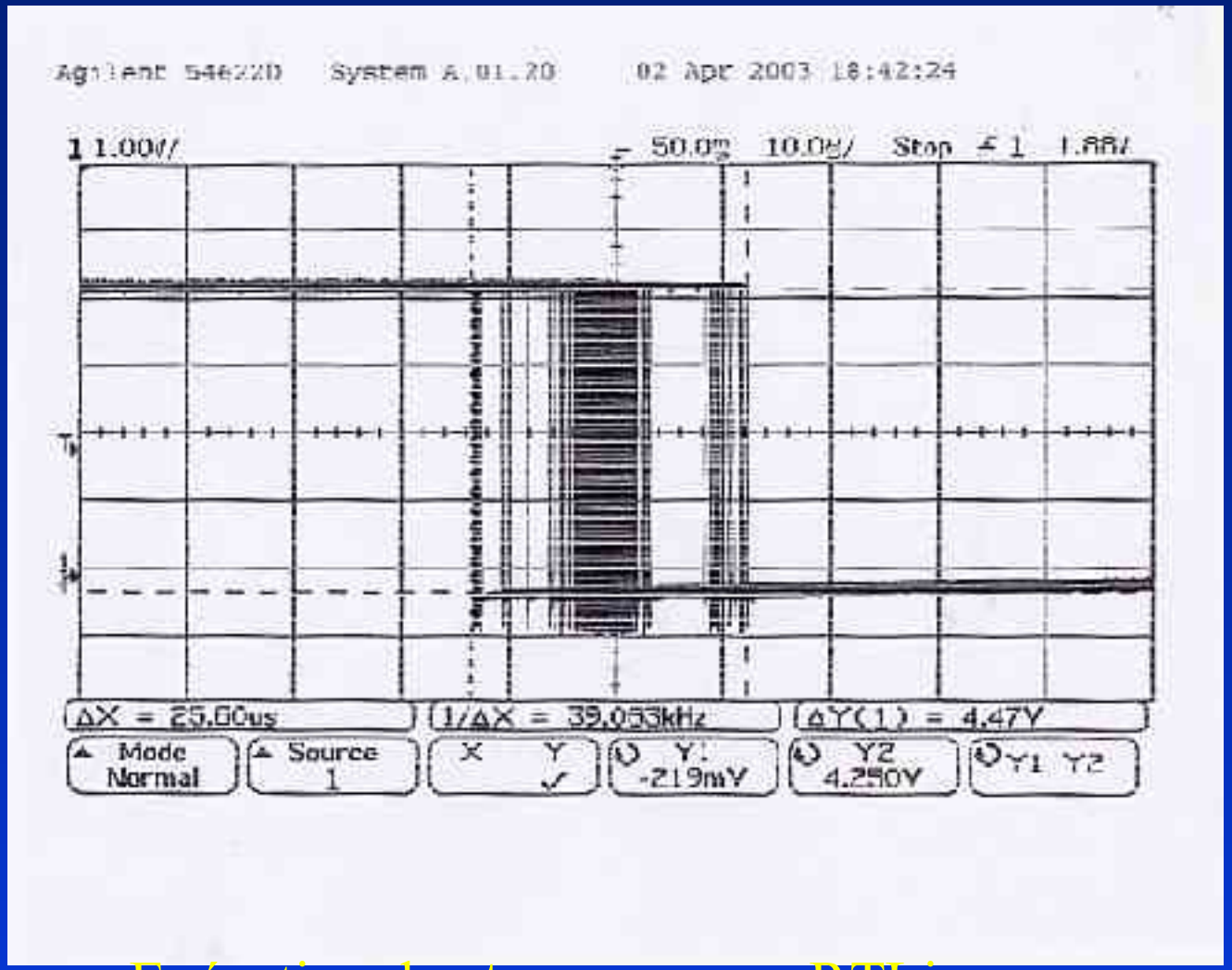


LA PETITE EXPERIENCE. SUITE ET FIN

- Mesure effectuée à l'oscilloscope lors de l'utilisation du programme rtsquare décrit précédemment modifié pour s ’exécuter comme tâche Temps Réel sous RTLinux.

Dans les conditions de stress du système (écriture continu d ’un fichier de 50 Mo), on obtient le résultat ci-dessous à l'oscilloscope. La mesure du jitter donne la valeur de 25.6 µs comparés aux 17.6 ms du noyau LINUX standard, ce qui correspond environ à un rapport 1000.

LA PETITE EXPERIENCE. SUITE ET FIN



Exécution de rtquare avec RTLinux

LA PETITE EXPERIENCE. SUITE ET FIN

```
#include <rtl_time.h>
#include <rtl_sched.h>
#include <asm/io.h>

/* Adresse du port parallèle */
#define LPT 0x378

/* Période de sollicitation de 50 ms */
int period=50000000;

/* Valeur envoyée sur le port parallèle */
int nibl=0x01;

/* Identifiant du thread applicatif */
pthread_t thread;
```

LA PETITE EXPERIENCE. SUITE ET FIN

```
/* Corps du thread applicatif */
void * bit_toggle(void *t)
{
    /* On programme un réveil de la tâche toute les 50 ms */
    pthread_make_periodic_np(thread, gethrtime(), period);

    while (1){
        /* Ecriture de la valeur sur le port // */
        outb(nibl,LPT);

        /* Calcul de la valeur suivante (négation) */
        nibl = ~nibl;

        /* Attente du réveil */
        pthread_wait_np();
    }
}
```

LA PETITE EXPERIENCE. SUITE ET FIN

```
int init_module(void)
{
    pthread_attr_t attr;
    struct sched_param sched_param;

    /* Priorité à 1 */
    pthread_attr_init (&attr);
    sched_param.sched_priority = 1;
    pthread_attr_setschedparam (&attr, &sched_param);

    /* Création du thread applicatif */
    pthread_create (&thread,  &attr, bit_toggle, (void *)0);

    return 0;
}
```

LA PETITE EXPERIENCE. SUITE ET FIN

```
void cleanup_module(void)
{
    pthread_delete_np (thread);
}
```

Programme TR rtsquare pour RTLinux (fichier C rtsquare.c)

BILAN

LE CHOIX D ’UN LINUX EMBARQUE TEMPS REEL

• Le choix est à faire en fonction des contraintes Temps Réel que doit respecter le système :

– Pas de contrainte. Best effort. Réactivité de qq 10ms à qq 100 ms.

–Temps Réel mou. Réactivité de qq 100 µs dans la très majorité des cas.

–Temps Réel dur. Réactivité de qq 10 µs (dans 100 % des cas).

LE CHOIX D 'UN LINUX EMBARQUE TEMPS REEL

	Non Temps Réel	Réactivité
Linux standard		qq 10-100 ms
Patch kpreempt Patch low lat Montavista Metrowerks TimeSys	Temps Réel mou	qq 100 µs
BlueCat RT Montavista RTLinux, RTAI	Temps Réel dur	qq 10 µs

LE CHOIX D ’UN LINUX EMBARQUE TEMPS REEL

• Le choix final se fera en fonction des contraintes temporelles imposées par le processus à contrôler depuis LINUX en prenant aussi en compte la complexité de mise en oeuvre dans chaque cas :

– Configuration du noyau.

– Ecriture de l'application temps réel.

• Choisir là aussi un linux embarqué Temps Réel commercial est rassurant. Cela a aussi un coût.

LE CHOIX D ’UN LINUX EMBARQUE TEMPS REEL

SIMPLE — **Complexité de programmation**

Linux standard	API Linux standard
Patch kpreempt Patch low lat Montavista Metrowerks	Application de patch au noyau Linux standard API Linux standard
TimeSys	
BlueCat RT Montavista	Application de patch au noyau Linux standard Installation de modules Linux spécifiques API spécifique ou POSIX
RTLinux, RTAI	**COMPLEXE**

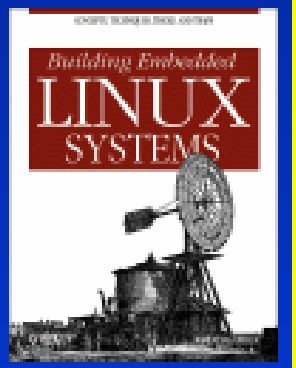
REFERENCES BIBLIOGRAPHIQUES

- Real-Time and Embedded Guide. H. Bruyninckx.
  http://people.mech.kuleuven.ac.be/~bruyninc/rthowto/
- Embedded Linux Howto
  http://linux-embedded.org/howto/Embedded-Linux-Howto.html
- Linux Magazine. Linux Temps Réel. Où en est-on aujourd ’hui ? P. Kadionik et Pierre Ficheux. Juillet-août 2003.
- http://www.enseirb.fr/~kadionik/embedded/linux_realtime/linuxrealtime.html
- La page de l ’auteur :
  http://www.enseirb.fr/~kadionik/embedded/embedded.html

REFERENCES BIBLIOGRAPHIQUES

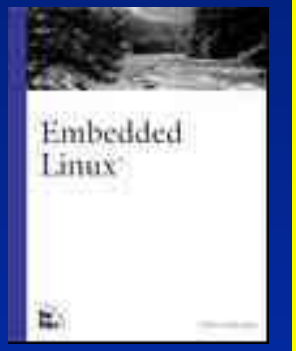
- **Linux embarqué. P. Ficheux. Editions Eyrolles.**
  LA REFERENCE !

- **Building Embedded Linux Systems. K. Yaghmour. Editions O' Reilly.**



REFERENCES BIBLIOGRAPHIQUES

- Embedded Linux. J. Lombardo. Editions New Riders.



- Embedded Linux. C. Hollabaugh. Editions Addison Wesley