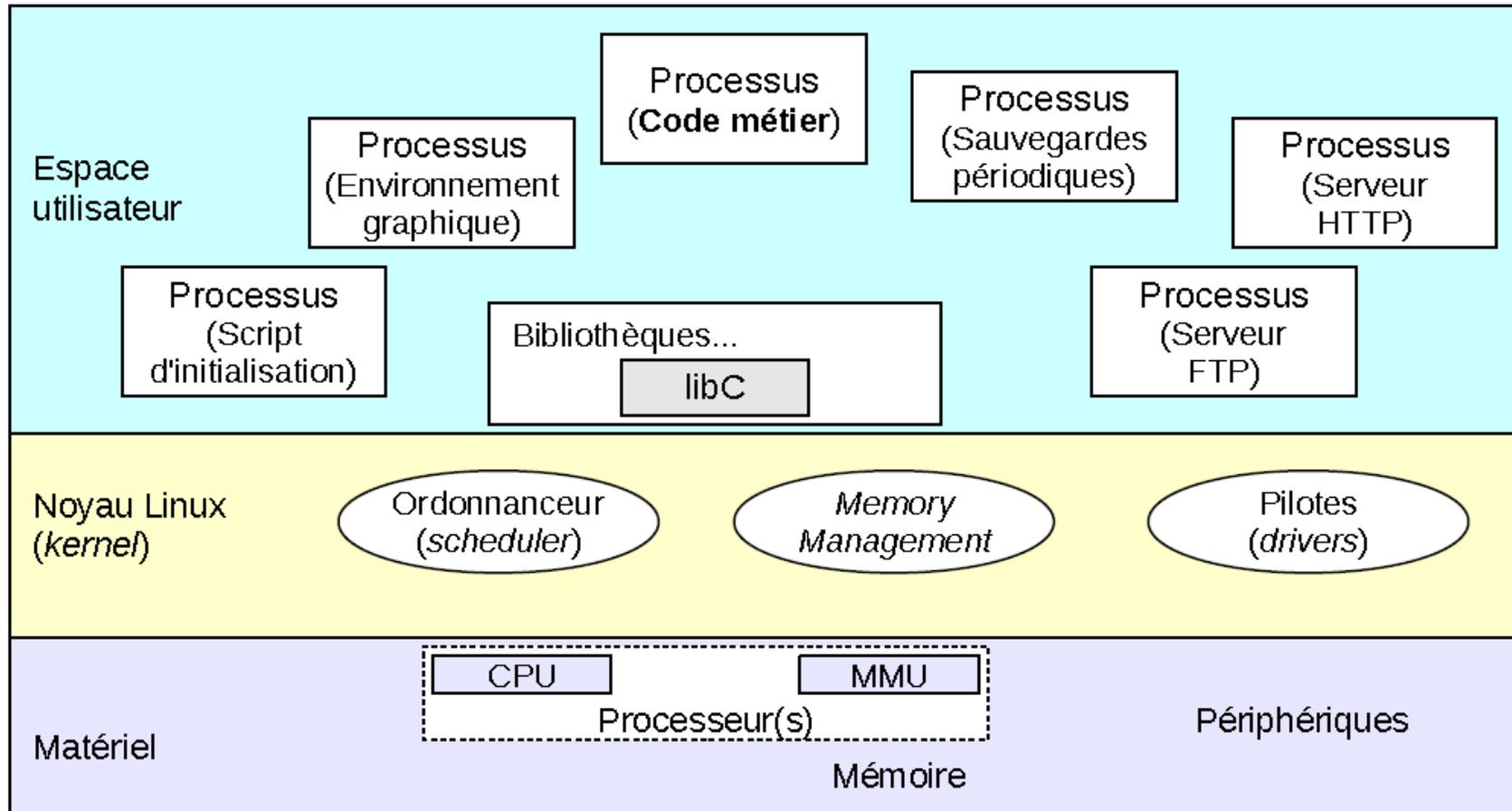


- Introduction
- Microcontrôleur / Microprocesseur
 - Terminologie
 - Comparaison de solutions
 - Choix d'une architecture
- Apports d'un système d'exploitation
 - Définition d'un OS
 - Abstraction des périphériques
 - Exécution des tâches
 - Mémoire virtuelle et MMU
- **Linux pour l'embarqué**
 - Pourquoi linux ?
 - Composants d'un système linux
 - Démarrage du système
 - Temps réel
 - Les principaux systèmes d'exploitation dans l'embarqué
 - Construire son système
 - Amélioration des performances du noyau linux

Pourquoi linux ?

- **Libre, disponible gratuitement au niveau source.**
- **Ouvert.**
- **Différentes distributions proposées suivant l'application :**
 - Téléphonie,
 - Routeur, switch, proxy, ...
 - Télévision,
 - Applications industrielles,
 - ...
- **Stable, performant.**
- **Support – communauté.**
- **Nombreux logiciels disponibles**
- **Connectivité Ip en standard.**
- **Portage sur tout type d'architecture (x86, ARM, MIPS, PowerPC, ...)**
- **Taille du noyau.**
- **Organisation modulaire = évolutivité**
- **Adaptation d'un spécialiste linux vers l'embarqué aisée**

Composants d'un système linux – Le noyau



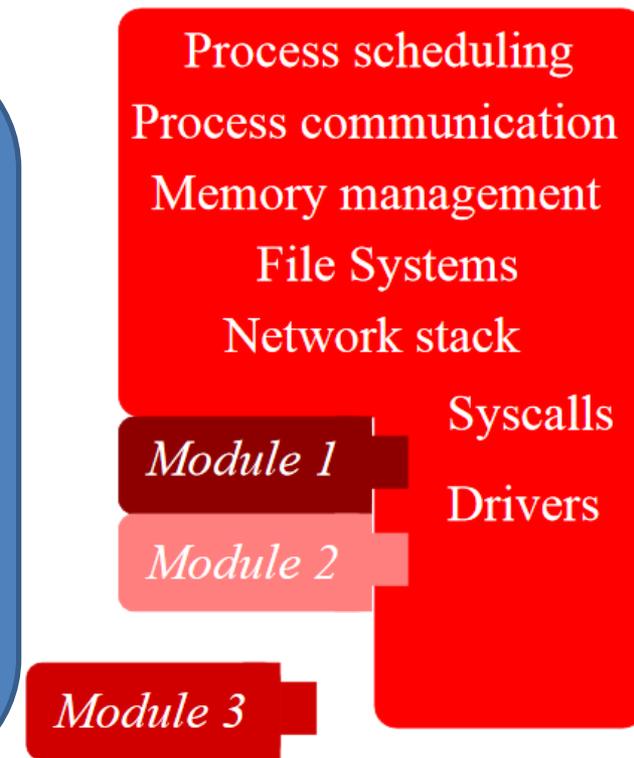
Rôle du noyau Linux : mettre les ressources matérielles à disposition des applications utilisateur. Le processeur exécute le code du noyau en mode superviseur (privilegié). Le processeur exécute le code en espace utilisateur en mode protégé (non-privilegié). Les tâches (threads) de l'espace utilisateur s'exécutent dans des processus (espaces de mémoire disjoints).

Composants d'un système linux – Les modules

- **Image du noyau (kernel image)** : un seul fichier créé après édition de liens des différents fichiers objets
 - Fichier chargé en mémoire au démarrage.
=> **Disponibilité des fonctionnalités incluses dès le démarrage du noyau**

Modules : compilation de certains éléments (pilotes de périphériques, systèmes de fichiers...)

- Chargement dynamique par le noyau en fonction des besoins
- Stockage de chaque module dans un fichier séparé (*.ko dans /lib/modules)
- Pas d'accès lors du démarrage initial



Composants d'un système linux – Configuration

- make menuconfig

```
Linux Kernel v2.6.16.60 Configuration

Linux Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters
are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded
<M> module < > module capable

Code maturity level options --->
General setup --->
Loadable module support --->
Block layer --->
Processor type and features --->
Power management options (ACPI, APM) --->
Bus options (PCI, PCMCIA, EISA, MCA, ISA) --->
Executable file formats --->
Networking --->
Device Drivers --->
File systems --->
Instrumentation Support --->
Kernel hacking --->
Security options --->
Cryptographic options --->
Library routines --->

Load an Alternate Configuration File
Save Configuration to an Alternate File

<Select> < Exit > < Help >
```

Composants d'un système linux – Configuration

- **make menuconfig**
- **Make :**
 - À saisir dans le répertoire racine des sources du noyau
 - Option -j <n> pour accélérer la compilation en utilisant plusieurs coeurs du processeur
 - Pas d'exécution nécessaire en tant que root
- **Génère :**
 - vmlinux : image non compressée du noyau mais non bootable
 - arch/<arch>/boot/*Image : image finale et bootable, généralement compressée, du noyau (**bzImage** pour x86, **zImage** pour ARM...)
 - Tous les modules du noyau, répartis dans l'arborescence des sources, sous la forme de fichiers .ko

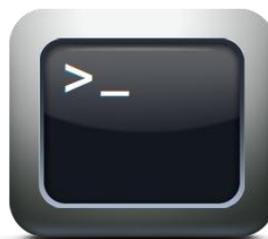
Composants d'un système linux – Shell

Shell : Couche logicielle qui fournit l'interface utilisateur d'un système d'exploitation.

Le shell d'un système d'exploitation peut prendre deux formes distinctes :

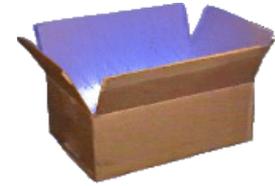
- Interface en ligne de commande (CLI)
- Shell graphique fournissant une interface graphique pour l'utilisateur (GUI, pour Graphical User Interface).

Dans les systèmes embarqués, l'utilisation d'une GUI est rare et non recommandée.



Composants d'un système linux – Busybox

- Logiciel libre souvent inclus dans l'embarqué
- Conçu comme un unique fichier exécutable
- Implémente un grand nombre des commandes standard sous Unix
- Configuration : **make busybox**



```
BusyBox 1.20.2 Configuration
----- Busybox Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < > module

Busybox Settings --->
--- Applets
  Archival Utilities --->
  Coreutils --->
  Console Utilities --->
  Debian Utilities --->
  Editors --->
  Finding Utilities --->
  Init Utilities --->
  Login/Password Management Utilities --->
  Linux Ext2 FS Progs --->
  Linux Module Utilities --->
  Linux System Utilities --->
  Miscellaneous Utilities --->
  Networking Utilities --->
  Print Utilities --->
  Mail Utilities --->
  Process Utilities --->
  Runit Utilities --->
  Shells --->
  System Logging Utilities --->
v(+)

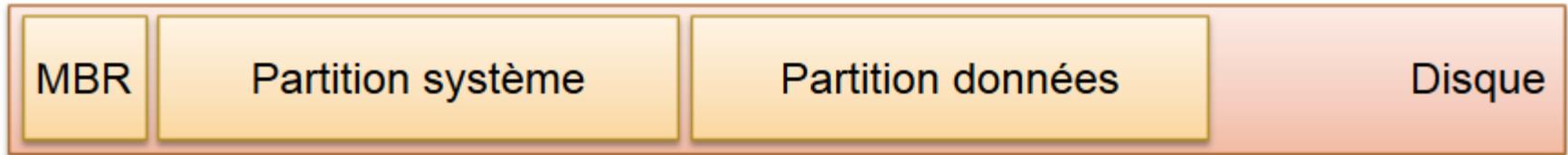
<Select> < Exit > < Help >
```

Composants d'un système linux – Système de fichier

Rep	description
/	Répertoire "racine", point d'entrée du système de fichiers
/boot	Contient le noyau Linux et l'amorceur
/bin	Contient les exécutables de base, comme par exemple cp, mv, ls, etc.
/dev	Contient des fichiers spéciaux nommés devices qui permettent le lien avec les périphériques de la machine
/etc	Contient les fichiers de configuration du système
/home	Contient les fichiers personnels des utilisateurs (un sous-répertoire par utilisateur)
/lib	Contient les bibliothèques et les modules du noyau (/lib/modules)
/media /mnt	Contient les « points de montage » des médias usuels : cd, dvd, disquette, clef usb
/opt	Lieu d'installation d'applications supplémentaires
/proc	Contient une "image" du système (/proc/kcore est l'image de la RAM)
/root	Répertoire personnel de l'administrateur
/sbin	Contient les exécutables destinés à l'administration du système
/tmp	Contient des fichiers temporaires utilisés par certains programmes
/usr	Contient les exécutables des programmes (/usr/bin et /usr/sbin), la documentation (/usr/doc) et les programmes pour le serveur graphique (/usr/X11R6).
/var	Répertoire contenant les fichiers qui servent à la maintenance du système (les fichiers de journaux notamment dans /var/log)

Composants d'un système linux – Système de fichier

- Support matériel : Disque dur, flash, SDcard, microSD, ...



- Le **M**aster **B**oot **R**ecord est situé dans les 1^{er} secteurs du disque
- Il est constitué de 2 parties :
 - La table des partitions
 - Le programme d'amorçage qui charge le noyau du système
- Plusieurs types de partitions
 - Principale
 - Etendue
 - Logique

Composants d'un système linux – Système de fichier

Chaque système est associé à un format de données

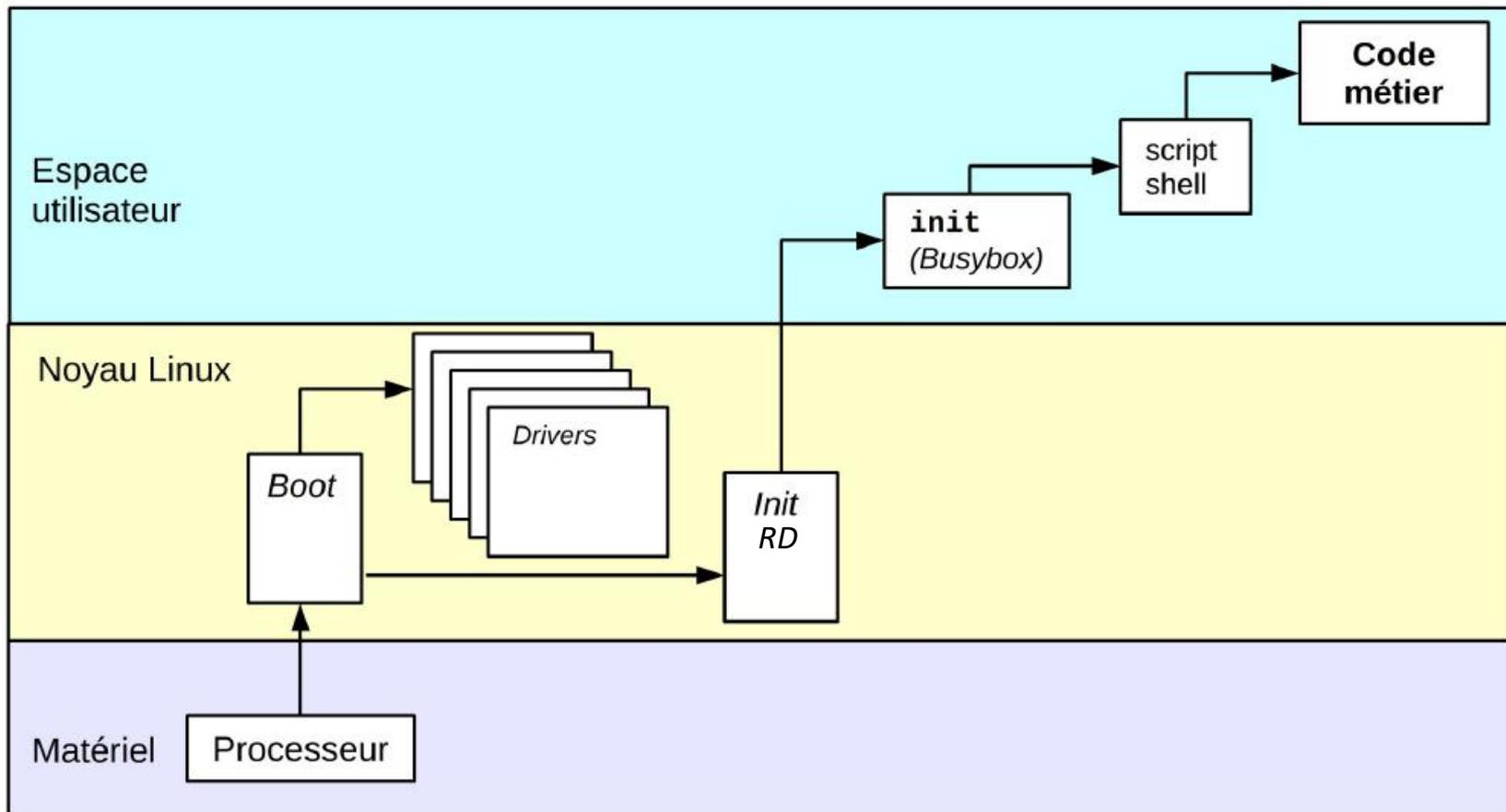
- **Sous Linux**
 - ext2, ext3, ext4, jfs, xfs, ...
 - ext2 non journalisé
- **Sous Windows**
 - fat, fat32, ntfs
 - Ntfs est utilisé sous Windows XP, Vista, Seven, Windows 8
- **Toujours préférer un système de fichier « journalisé » (si possible)**
 - Chaque séquence de lecture/écriture est d'abord inscrite dans un journal avant d'être effectuée
 - Si le système se bloque pendant la séquence, elle sera achevée après le redémarrage

=> On évite les erreurs dans le système de fichiers

Démarrage du système - Boot

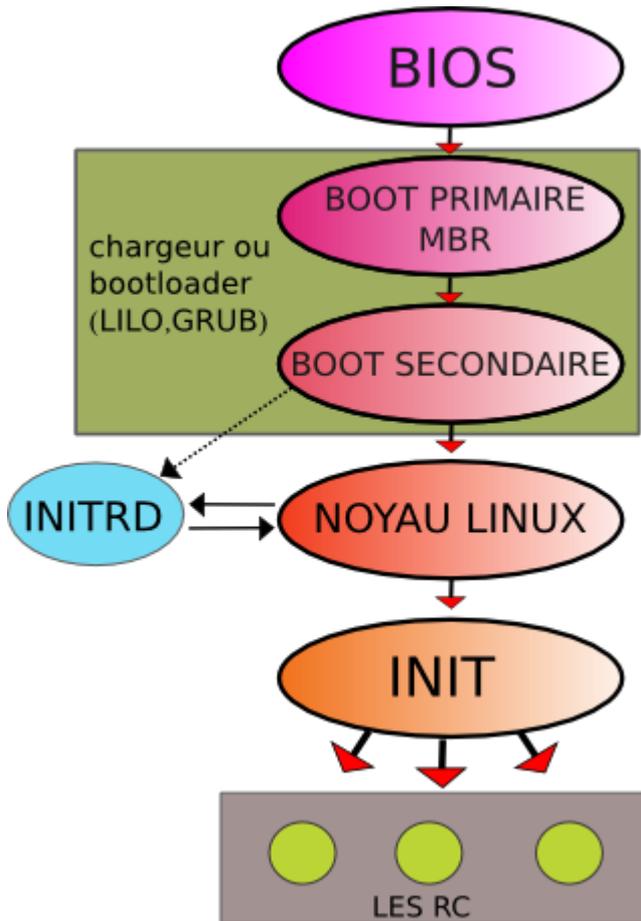
Durée de la séquence de boot :

- Chargement du noyau : **2 à 5 s**
- Processus init (montage du FS, config. paramètres /proc) : **1 à 2 s**
- Lancement des services (réseau, authentification, GUI, ...) : **>10 s**



Démarrage du système - Boot

Détail de la séquence de boot :



- **Bootloader** : Lilo, Syslinux, Grub, **U-Boot**, **BareBox**, ...
- **initrd** (INITIAL RamDisk) : image du noyau minimal initialisé au démarrage du système et indique où trouver le FS (ex : root=/dev/sdb2)
- **init** : Gère tous les processus (PID=1) y compris le noyau (kthread : PID=2)
- **inittab** : Fichier de configuration de init situé dans /etc. Définit les niveaux d'exécution et les terminaux virtuels (tty)
- **Scripts de rcx.d** : (x=0 à 6) contiennent les scripts de démarrage des services suivant les niveaux d'exécution (compatible POSIX)

Démarrage du système - Initialisation

Plusieurs versions de l'initialisation du système :

- **BSD Unix** : un seul script de démarrage principal `/etc/rc` qui appelle les scripts `/etc/rc.d/*` et `/etc/rc.local`
- **System V** : basé sur le concept de niveaux d'exécutions.
 - `/etc/inittab`
 - `/etc/rc.d/rc.sysinit`
 - `/etc/rc.d/rcx.d/*` ($x=0..6$)

```
$ ls /etc/rc.d/rc5.d
K15httpd      S00microcode_ctl
K20nfs        S04readahead_early
K28amd        S06cpuspeed
K50netconsole S08arpables_jf
```

- Lance 6 consoles virtuelle (tty1 à 6) et l'interface graphique X11

Au lieu de `/etc/rc.d/rc.sysinit`, les OS de la famille Debian utilisent `/etc/init.d/rcS` qui exécute les scripts `/etc/rcS.d/S*` dans l'ordre.

Level	Purpose	
	Most Linux	Some Linux
0	Shut down and power off	Shut down and power off
1	Single-user mode	Single-user mode
2	Multi-user console login, no networking	Multi-user console login, networking enabled
3	Multi-user console login, networking enabled	Multi-user graphical login, networking enabled
4	<i>not used</i>	<i>not used</i>
5	Multi-user graphical login, networking enabled	<i>not used</i>
6	Shut down and reboot	Shut down and reboot

Démarrage du système - Initialisation

Plusieurs versions de l'initialisation du système :

- **SystemD** : démon de gestion système
 - parent de tous les autres processus, directement ou indirectement => **PID = 1**
 - collection de logiciels sous forme de binaires
 - pilotage par un fichier texte d'une dizaine de lignes
 - liens symboliques dans **/etc/systemd/system/** vers les fichiers situés dans **/usr/lib/systemd/system/**.
 - Utilisation de la commande **systemctl**

```
test@test-virtual-machine:~$ pstree
systemd--ManagementAgent--2*[{CThreadUtils::s}]
      |
      |--ModemManager--[{gdbus}]
      |                  |
      |                  |--{gmain}
      |
      |--NetworkManager--dhclient
      |                   |
      |                   |--dnsmasq
      |                   |
      |                   |--[{gdbus}]
      |                   |
      |                   |--{gmain}
      |
      |--VGAAuthService
      |
      |--accounts-daemon--[{gdbus}]
      |                  |
      |                  |--{gmain}
```

```
# systemctl start [name.service]
# systemctl stop [nom.service]
# systemctl restart [nom.service]
# systemctl reload [nom.service]
$ systemctl status [nom.service]
```

Linux et le temps réel

- **Temps réel souple (soft realtime)**
 - Contraintes temporelles en millisecondes
 - Comportement moyen, pas de garantie (best effort)
 - Système d'ordonnancement performant : CFS (Completely Fair Scheduler) depuis la version 2.6.23
 - **Linux vanilla**
 - Contraintes temporelles en centaines de microsecondes
 - Comportement prévu pour gérer les pires circonstances (worst cases)
 - **Linux avec patch PREEMPT_RT (Ingo Molnar)**

Linux et le temps réel

- **Temps réel strict (hard realtime)**
 - **Qualifié mais non-certifié**
 - Contraintes temporelles en dizaines de microsecondes
 - Comportement dans le pire des cas vérifiable en pratique mais pas prouvable à cause des millions de lignes de code du noyau Linux sous-jacent.
 - **Linux avec extension Xenomai / Adeos**
 - **Certifié**
 - Contraintes temporelles en microsecondes
 - Comportement vérifiable (code minimal)
 - **RTEMS** (Real-Time Executive for Multiprocessor Systems)
 - **FreeRTOS**

Construire son système : Réduire l'empreinte mémoire !

Mémoire est très couteuse => 3 façons de réduire l'empreinte mémoire:

1. Optimiser le noyau

- Enlever le code dont on n'a pas besoin
- Optimiser la compilation -
- Enlever le swap
- Voir le «Linux tiny kernel project »: optimisations sous forme de patch.

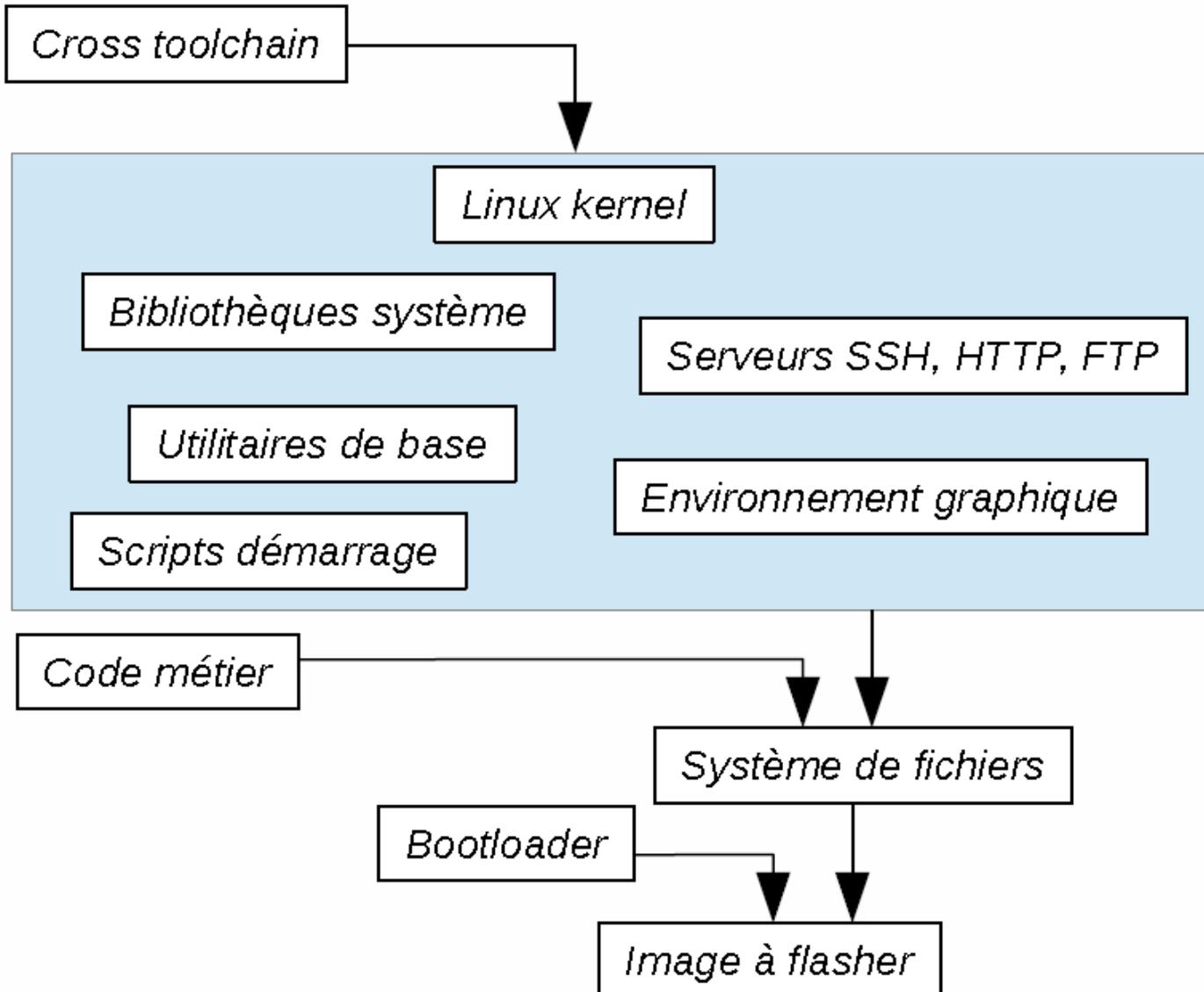
2. Optimiser l'espace de l'applicatif

- Optimiser son code
- Optimiser l'utilisation des bibliothèques
 - Bibliothèques partagées dans les applications
 - Utiliser des bibliothèques réduites (uClibc, diet libc, ...)
- Utiliser des applications optimisées : BusyBox, TinyLogin, Serveur web BOA, mini_httpd, GoAhead, ...

3. Compresser le système de fichiers

- Certains systèmes de fichiers sont compressés: JFFS2, CRAMFS

Construire son système



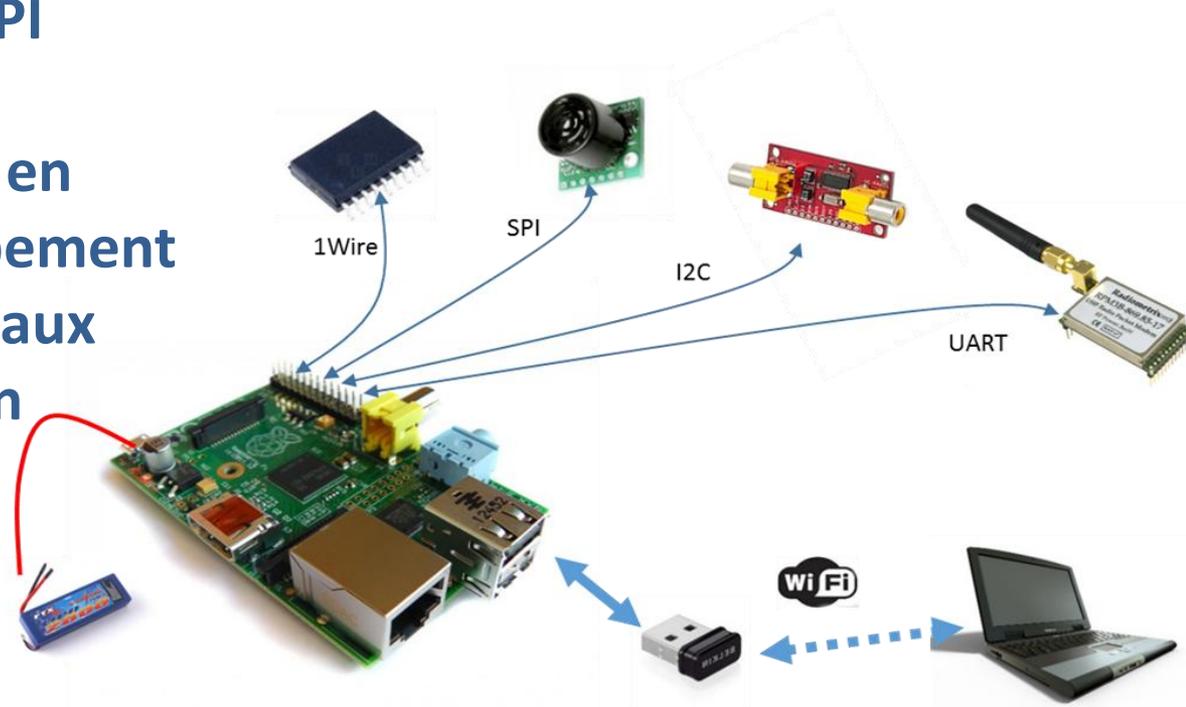
Construire son système – Différentes solutions

	Pour	Contre
Tout construire manuellement	<ul style="list-style-type: none"> • Personnalisation complète • Maîtrise complète du système • On acquiert de l'expérience 	<ul style="list-style-type: none"> • Gérer les dépendances (« c'est l'enfer ») • Besoin de connaître tous les détails du système • Compatibilité des versions • Manque de reproductibilité
Distributions installables (Debian, Ubuntu, openSuse, ...)	<ul style="list-style-type: none"> • Facile à installer et à administrer 	<ul style="list-style-type: none"> • Difficile à personnaliser • Difficile à optimiser (temps de démarrage, la taille) • Difficile de reconstruire le système complet à partir des sources • « Grand système » • Beaucoup de dépendances • Non disponible pour toutes les architectures
Build systems Buildroot, Yocto, PTXdist, etc	<ul style="list-style-type: none"> • Beaucoup de souplesse • Construction à partir des sources : personnalisation et optimisation faciles • entièrement reproductible • Utilise la compilation croisée • Paquets spécifiques pour l'embarqué • Fonctionnalités en option 	<ul style="list-style-type: none"> • Pas aussi facile une distribution binaire • Le temps de construction (compilation)

Les principaux systèmes d'exploitation dans l'embarqué – linux

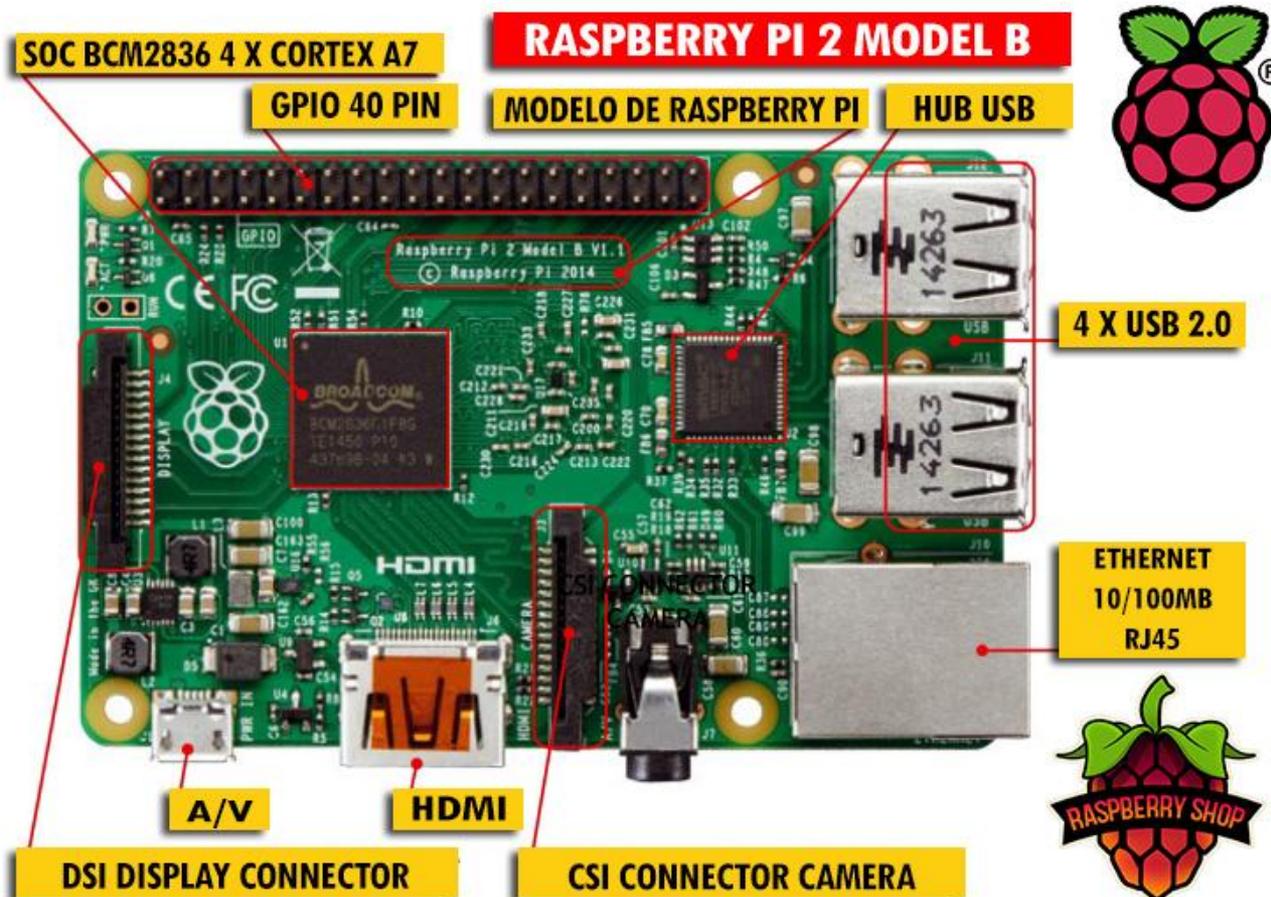
TP : Prise main d'un système embarqué Raspberry Pi

- Prise en main d'un système embarqué Raspberry Pi
- Distribution Raspbian
- Les pratiques de mise en œuvre et de développement
- Utilisation des principaux bus de communication



Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi



Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

i2c (pins 03, 05)
w1 (pin 07)
GPIO (pins 19, 21, 23)
UART (pins 08, 10)
SPI (pins 24, 26)

Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

GPIO : Global Purpose Input Output

```
$ cd /sys/class/gpio
/sys/class/gpio $ ls
export      gpiochip0  unexport

/sys/class/gpio $ echo 24 > export
/sys/class/gpio $ ls
export      gpio24      gpiochip0  unexport

/sys/class/gpio $ cd gpio24/
/sys/class/gpio/gpio24 $ ls
active_low  direction  edge       subsystem  uevent     value
```

Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

GPIO : Global Purpose Input Output

Accès en écriture :

```
/sys/class/gpio/gpio24 $ echo out > direction  
/sys/class/gpio/gpio24 $ echo 1 > value  
/sys/class/gpio/gpio24 $ echo 0 > value
```

Accès en lecture :

```
/sys/class/gpio/gpio24 $ echo in > direction  
/sys/class/gpio/gpio24 $ cat value  
/sys/class/gpio/gpio24 $ 0
```

Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

Bus i2c : inclure les drivers dans le noyau :

```
(linux sources dir) $ make menuconfig
```

```
Device drivers --->
  I2C support --->
    <*> I2C support
      [*] Enable compatibility bits for old user-space
    <*> I2C device interface
    <*> I2C bus multiplexing support
      Multiplexer I2C Chip support --->
    [*] Autoselect pertinent helper modules
      I2C Hardware Bus support --->
        <*> Atmel AT91 I2C Two-Wire interface (TWI)
        <*> GPIO-based bitbanging I2C
```

Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

Bus i2c : Ajouter le module au noyau après son chargement :

```
$ sudo modprobe i2c-dev
$
$ ls /dev/i2*

/dev/i2c-0      (réservé)
/dev/i2c-1
```

Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera          Enable/Disable connection to the
P2 SSH             Enable/Disable remote command lin
P3 VNC            Enable/Disable graphical remote a
P4 SPI            Enable/Disable automatic loading
P5 I2C            Enable/Disable automatic loading
P6 Serial         Enable/Disable shell and kernel m
P7 1-Wire         Enable/Disable one-wire interface
P8 Remote GPIO    Enable/Disable remote access to G

<Select>          <Back>
```

Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

Bus i2c : Utiliser dans le terminal :

```
$ sudo apt-get install i2ctools
$
$ i2cdetect -> Détection des périphériques
$ i2cget -> Lire
$ i2cset -> Ecrire
```

Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

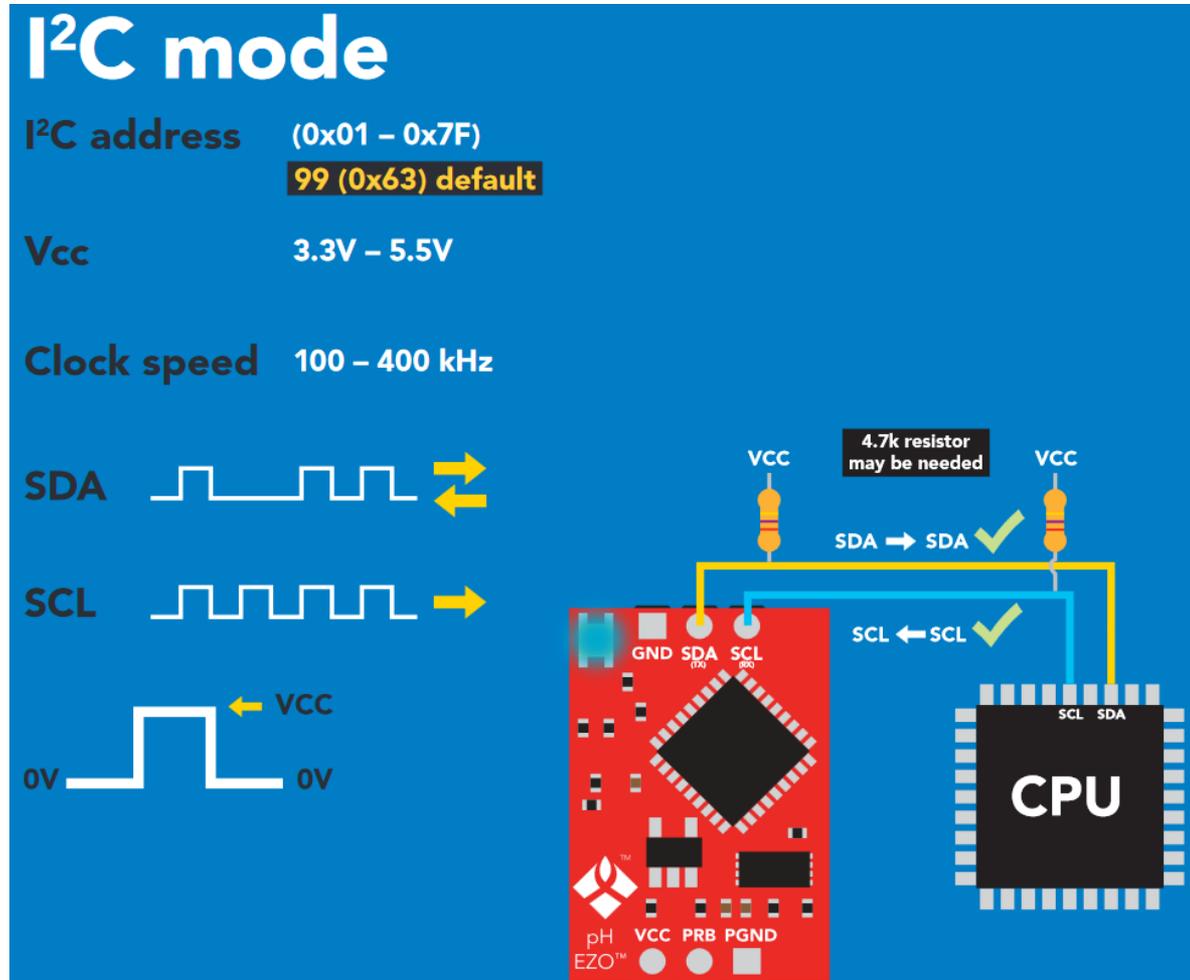
Bus i2c : Utiliser dans le terminal :

```
pi@raspberrypi: ~  
login as: pi  
pi@192.168.1.20's password:  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon May  9 20:12:11 2016 from dominic-pc  
pi@raspberrypi:~ $ i2cdetect -y 1  
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
pi@raspberrypi:~ $
```

127
adresses

Les principaux systèmes d'exploitation dans l'embarqué – linux

Bus i2c : exemple



Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Prise main d'un système embarqué Raspberry Pi

Bus i2c : Utiliser dans le terminal :

```
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.
```

```
Last login: Sun Jul 17 11:27:16 2016 from dominic-pc
```

```
pi@raspberrypi:~ $ i2cdetect -y 1
```

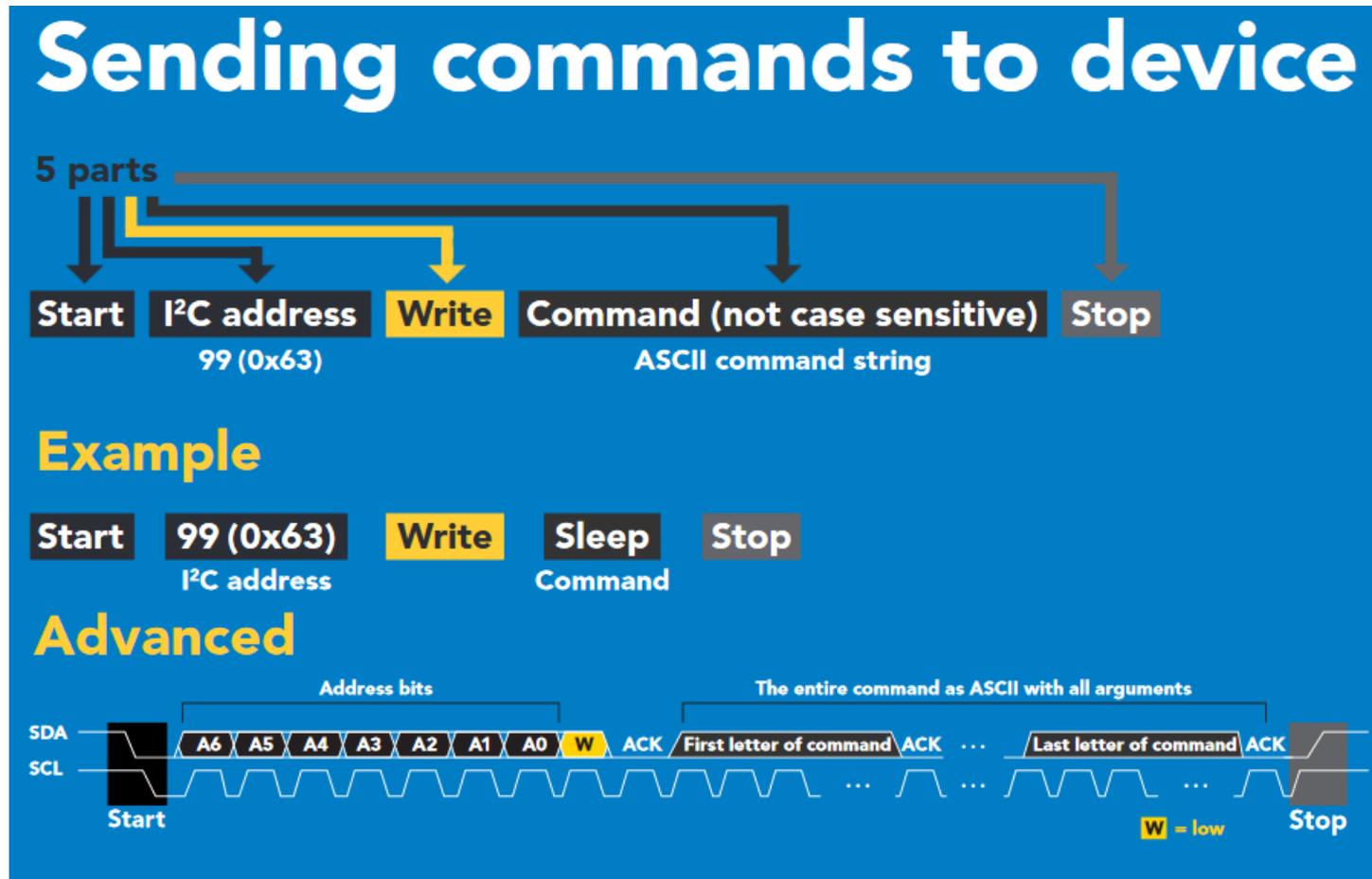
```
 0 1 2 3 4 5 6 7 8 9 a b c d e f  
00: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
20: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
30: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
40: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
50: -- -- -- -- -- -- -- -- -- -- -- -- -- --  
60: -- -- 62 63 64 -- 66 -- -- -- -- -- -- -- --  
70: -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Adresses des périphériques
détectés

```
pi@raspberrypi:~ $ █
```

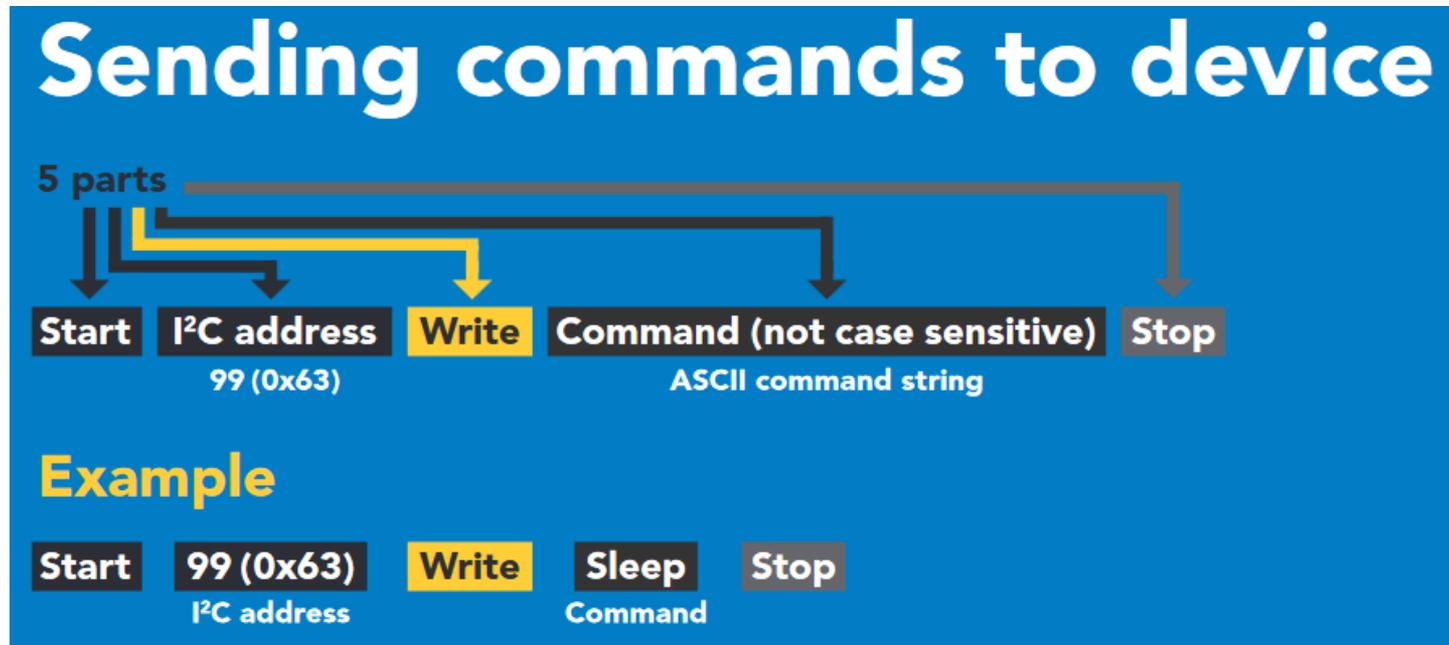
Les principaux systèmes d'exploitation dans l'embarqué – linux

Bus i2c : exemple



Les principaux systèmes d'exploitation dans l'embarqué – linux

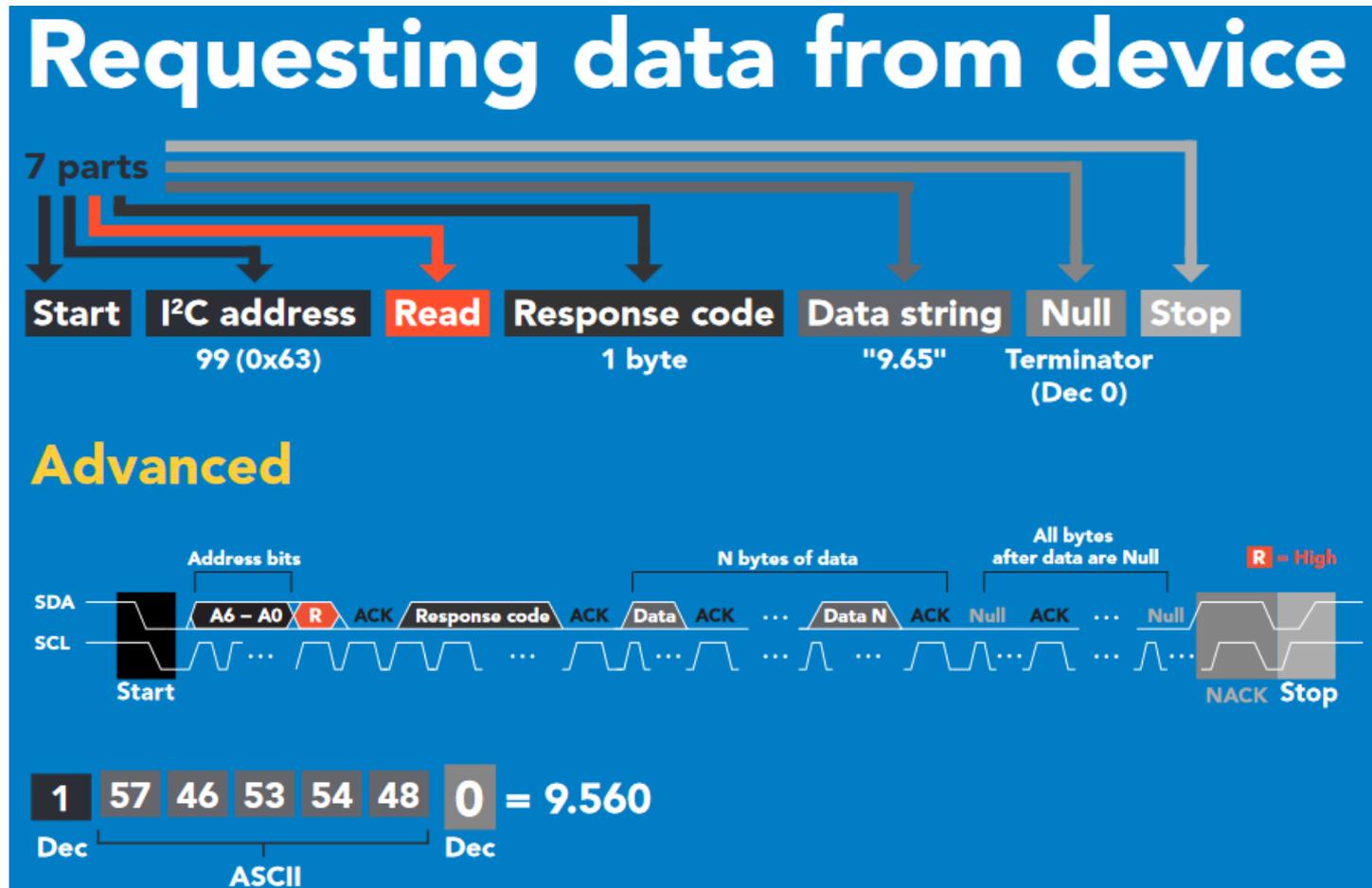
Bus i2c : exemple



```
$ i2cset 0x63 0x53 0x6c 0x65 0x65 0x70 i
```

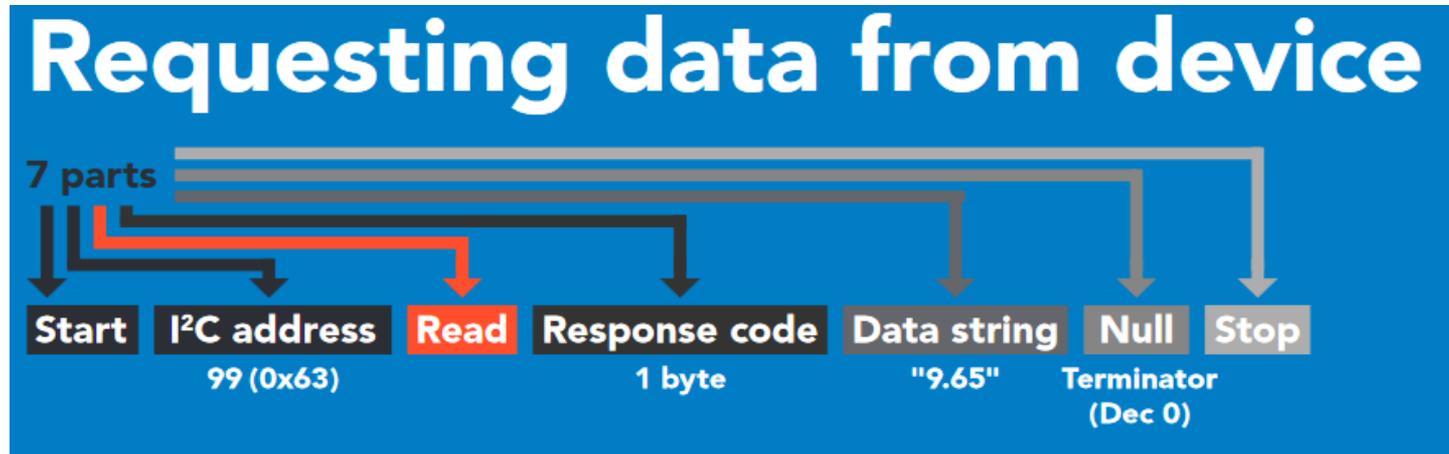
Les principaux systèmes d'exploitation dans l'embarqué – linux

Bus i2c : exemple



Les principaux systèmes d'exploitation dans l'embarqué – linux

Bus i2c : exemple

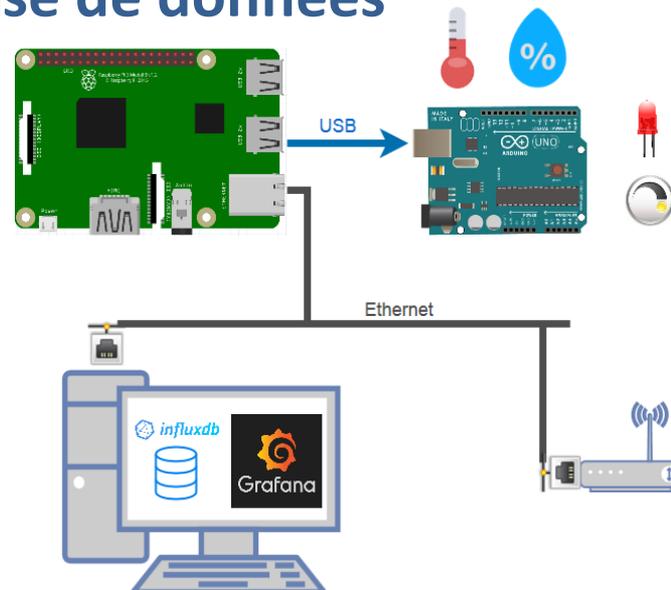


```
$ i2cget 0x63 0 5  
0x01392e363300
```

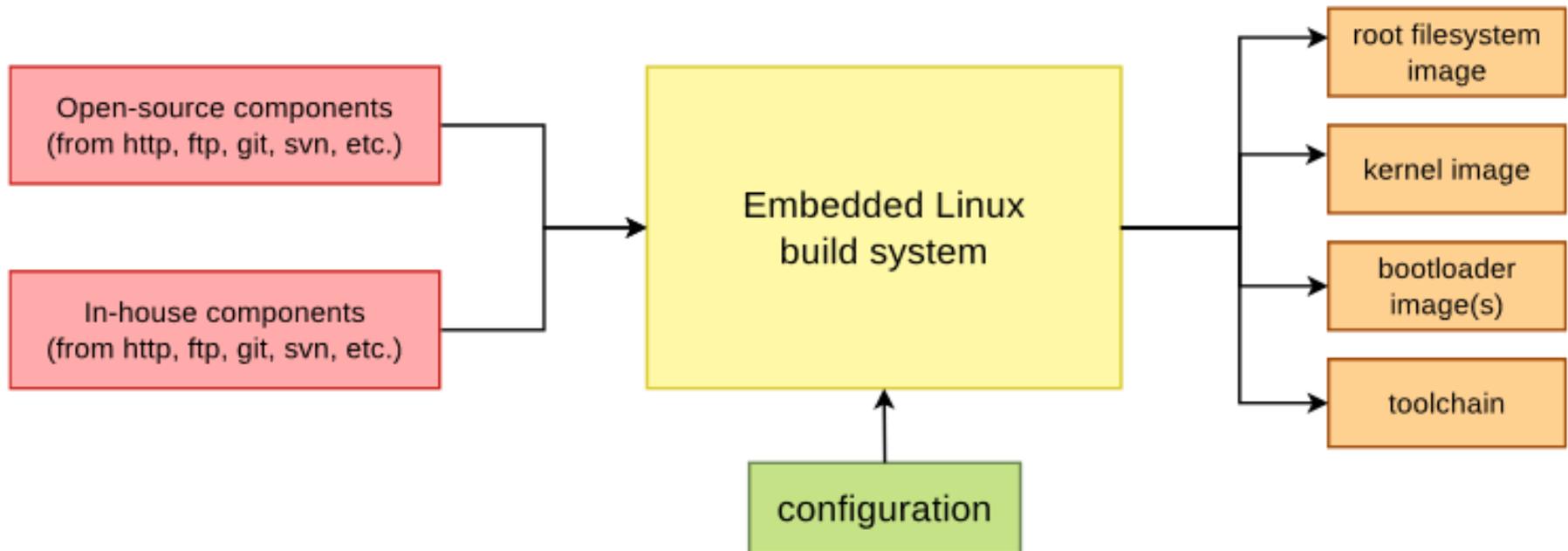
Les principaux systèmes d'exploitation dans l'embarqué – linux

TP : Objet connecté

- Programmation Microcontrôleur
- Liaison série asynchrone – ports IO – entrées analogiques
- Architecture IOT
- Panneau de contrôle (dashboard) Node Red
- Gestion d'une base de données



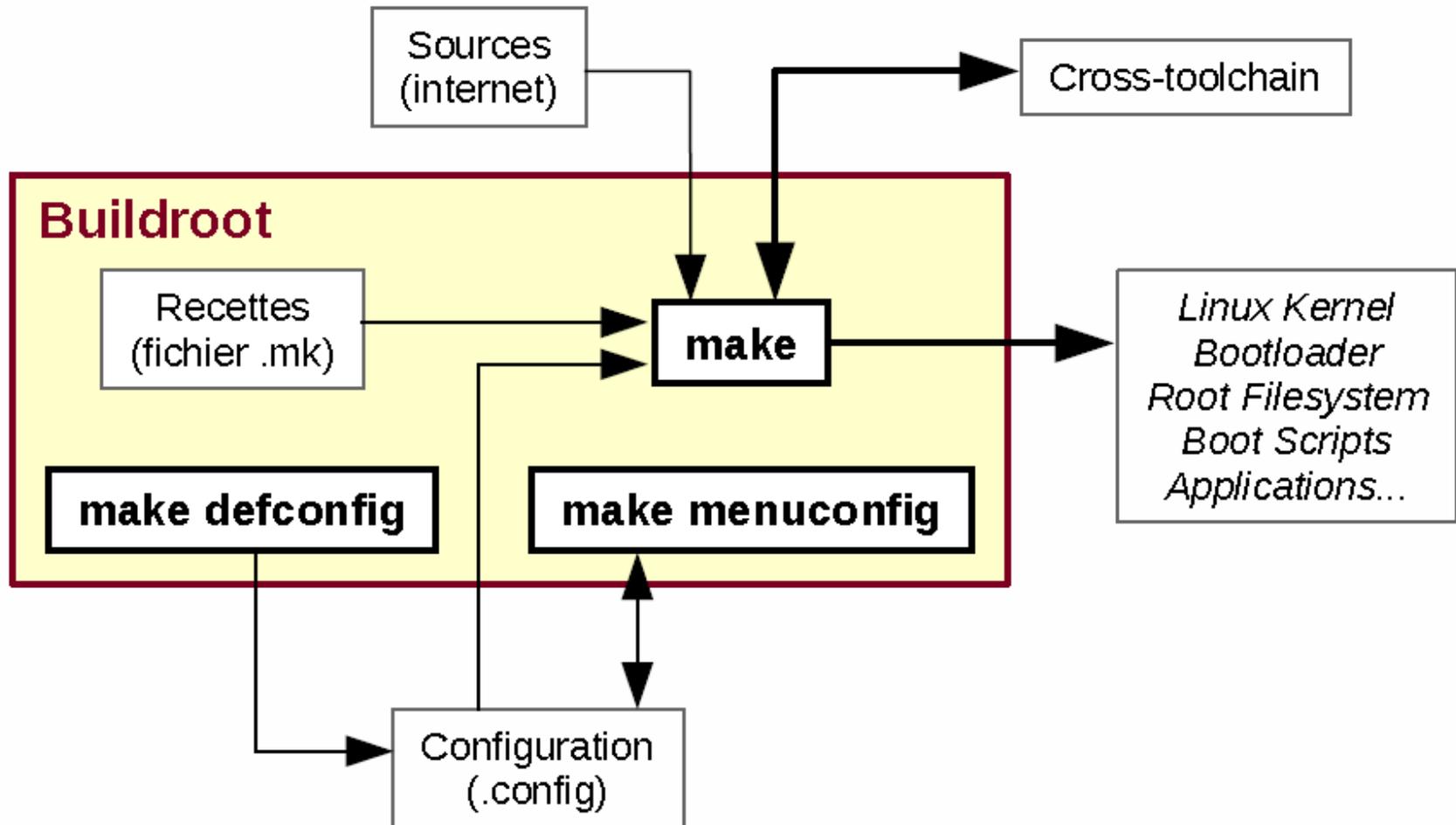
Construire son système – Principe d'un Build System



- **Compilation des sources** => beaucoup de flexibilité
- **Cross-compilation** => machines plus performantes
- **Recettes de construction** => facile

Construire son système – Buildroot

Buildroot est un ensemble de scripts et de fichiers de configuration permettant la construction complète d'un système Linux pour une cible embarquée. Il télécharge automatiquement les paquetages nécessaires pour la compilation et l'installation.



Construire son système - Buildroot

make menuconfig

```
Buildroot 2015.02 Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a
feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] feature is selected [ ] feature is

Target options --->
Build options --->
Toolchain --->
System configuration --->
Kernel --->
Target packages --->
Filesystem images --->
Bootloaders --->
Host utilities --->
Legacy config options --->

<Select> < Exit > < Help > < Save > < Load >
```

Construire son système - Buildroot

make defconfig

- **Buildroot est livré avec un certain nombre de configurations de bases pour diverses plates-formes matérielles :**
 - RaspberryPi,
 - BeagleBone Black,
 - CubieBoard,
 - Atmel evaluation boards,
 - Divers i.MX6 boards,
 - ...
- **Construit seulement un système minimum :**
 - toolchain,
 - bootloader,
 - Kernel
 - root filesystem minimum

Construire son système – Buildroot : Fonctionnement globale

▶ output/

▶ images/

- ▶ zImage
- ▶ armada-370-mirabox.dtb
- ▶ rootfs.tar
- ▶ rootfs.ubi

**Boot partition
FAT16**

**Rootfs partition
EXT4**

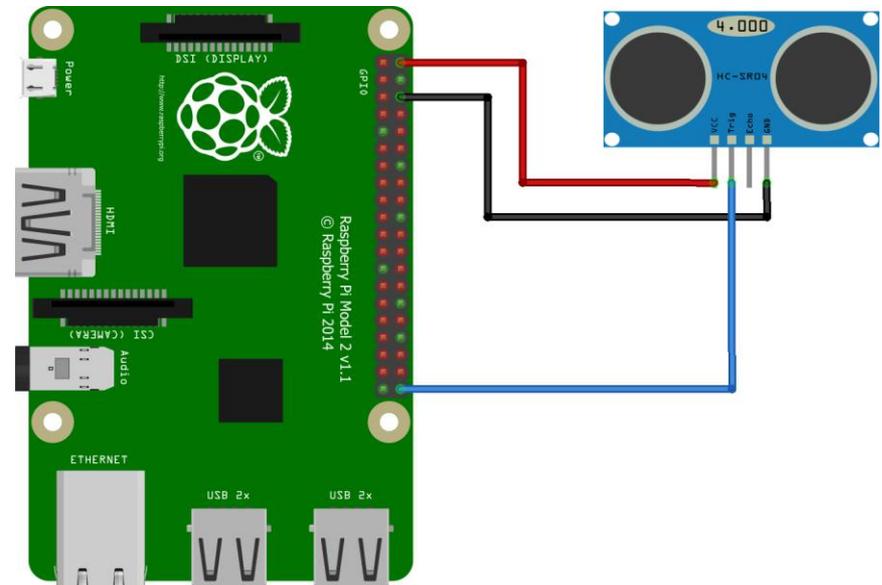


Construire son système – Buildroot

TP : Construction d'un système linux embarqué complet



- Noyau Linux
- Système de fichiers racine
- Utilitaires de base (Busybox)
- Code métier C++
- Production d'un signal
- Performances temps réel



Amélioration des performances du noyau linux

Rappel et définition

**Le temps-réel est une notion de garantie
et non pas de performance**

- **qualité de fonctionnement (stabilité)**
- **temps de réponse : réaction appropriée → en un temps borné
→ à un événement**
- **Le temps réel est lié à des processus sensibles
(militaire, spatial, énergie, médical, transport, ...)**

Amélioration des performances du noyau linux

Mesures des performances temps réelles sur linux

Principe :

- Programmer une tâche périodique
- Comparer la date d'échéance théorique avec la date réelle
- La différence correspond au « jitter » (gigue)
- La gigue est caractéristique de la « latence » du système
- Tester le système avec ET sans charge !

Outils :

- Tâche périodique: `cyclictest`, `latency`
- Manipulation de « GPIO »
- Stimulation: `hackbench`, `stress`, `dohell`
- Mesure : Oscilloscope, Gnuplot (tracé de courbes)

Amélioration des performances du noyau linux

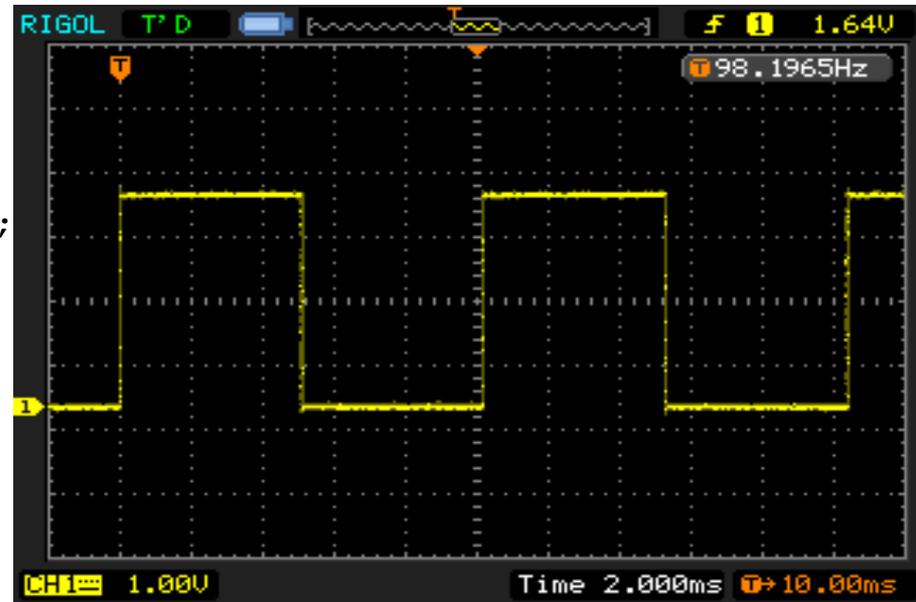
Mesures des performances temps réelles sur linux

```
squareSignal.cpp

int duration = atoi(argv[1]);

ofstream fs("/sys/class/gpio/gpio21/value");

while(1)
{
    fs << "1" << endl;
    usleep(duration);
    fs << "0" << endl;
    usleep(duration);
}
```

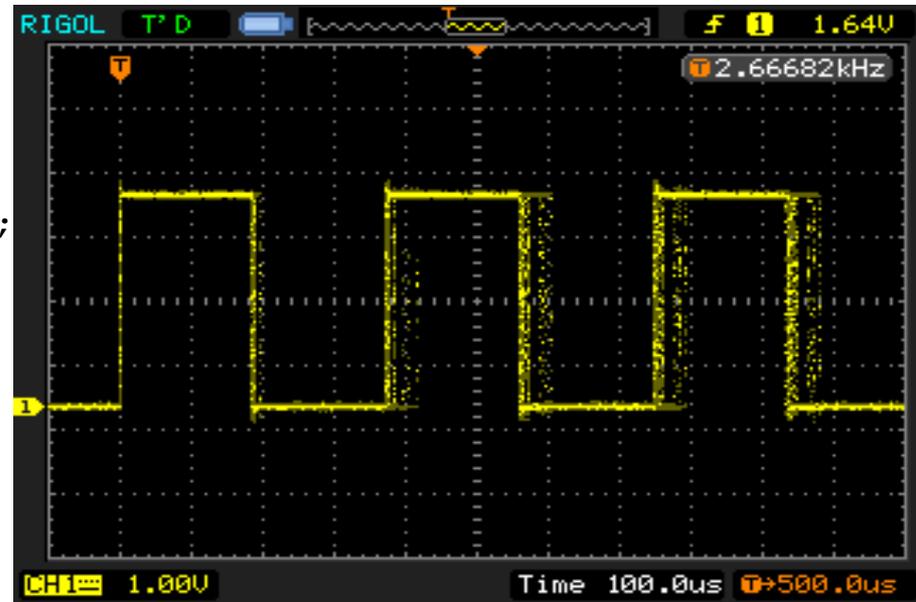


squareSignal 5000

Amélioration des performances du noyau linux

Mesures des performances temps réelles sur linux

```
squareSignal.cpp  
  
int duration = atoi(argv[1]);  
  
ofstream fs("/sys/class/gpio/gpio21/value");  
  
while(1)  
{  
    fs << "1" << endl;  
    usleep(duration);  
    fs << "0" << endl;  
    usleep(duration);  
}
```



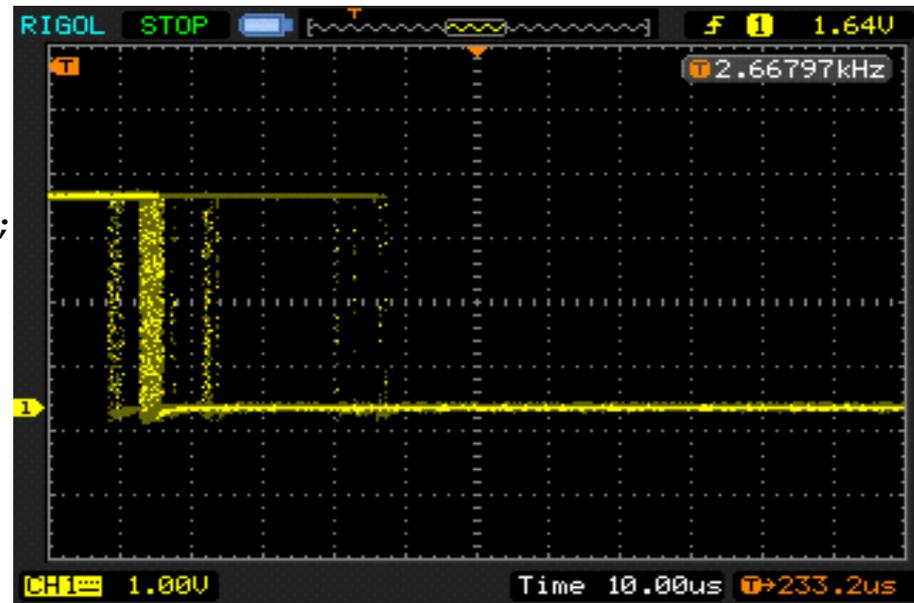
squareSignal 100

Activation de la persistance de l'affichage

Amélioration des performances du noyau linux

Mesures des performances temps réelles sur linux

```
squareSignal.cpp  
  
int duration = atoi(argv[1]);  
  
ofstream fs("/sys/class/gpio/gpio21/value");  
  
while(1)  
{  
    fs << "1" << endl;  
    usleep(duration);  
    fs << "0" << endl;  
    usleep(duration);  
}
```



squareSignal 100

Mesure des retards/latences

Amélioration des performances du noyau linux

Mesures des performances temps réelles sur linux

```
RT_TASK sqrtSig_task;

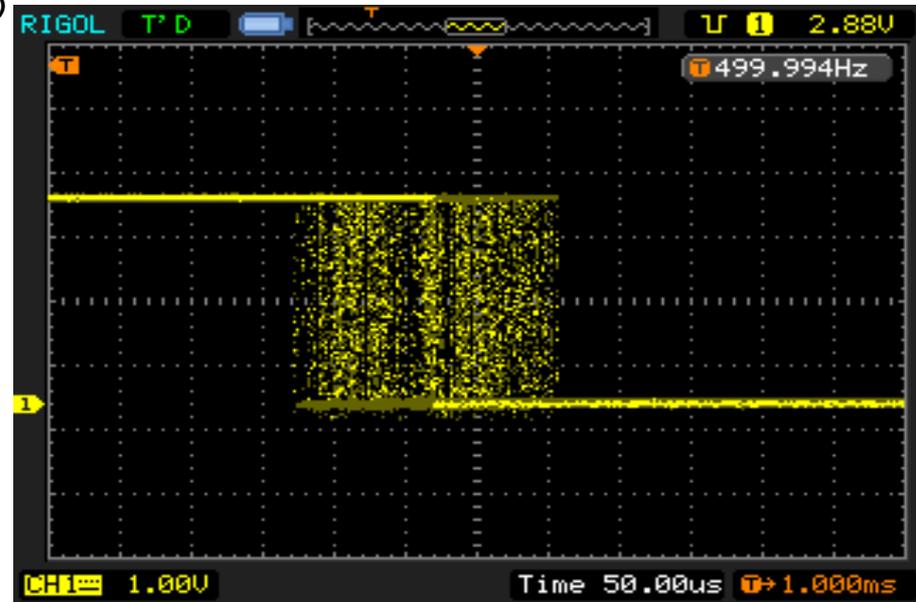
void sqrtSig(void *arg __attribute__((__unused__)))
{
    rt_task_set_periodic(NULL, TM_NOW, TIMESLEEP);
    while(1) {
        rt_task_wait_period(NULL);
        ...
    }
}

int main(void)
{
    ...
    printf("Create task sqrSig \n");
    rt_task_create(&blink_task,"sqrSig",0,99,0);

    printf("Start task sqrSig ");
    rt_task_start(&sqrSig_task,&sqrSig,NULL);

    printf("Press Enter to stop task\n");
    getchar();

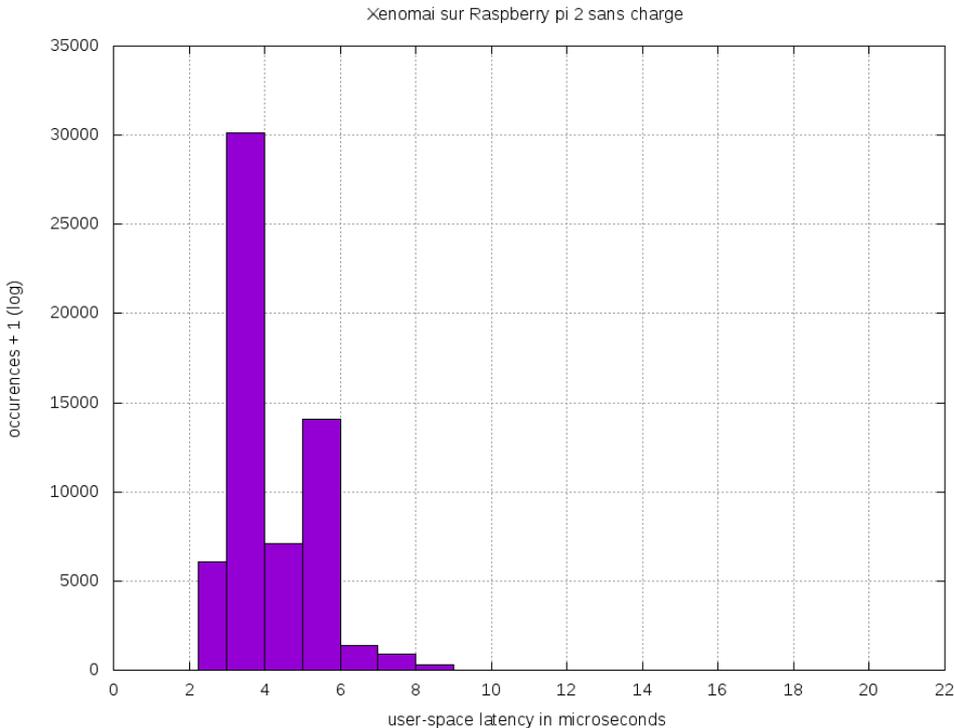
    rt_task_delete(&sqrSig_task);
    ...
}
```



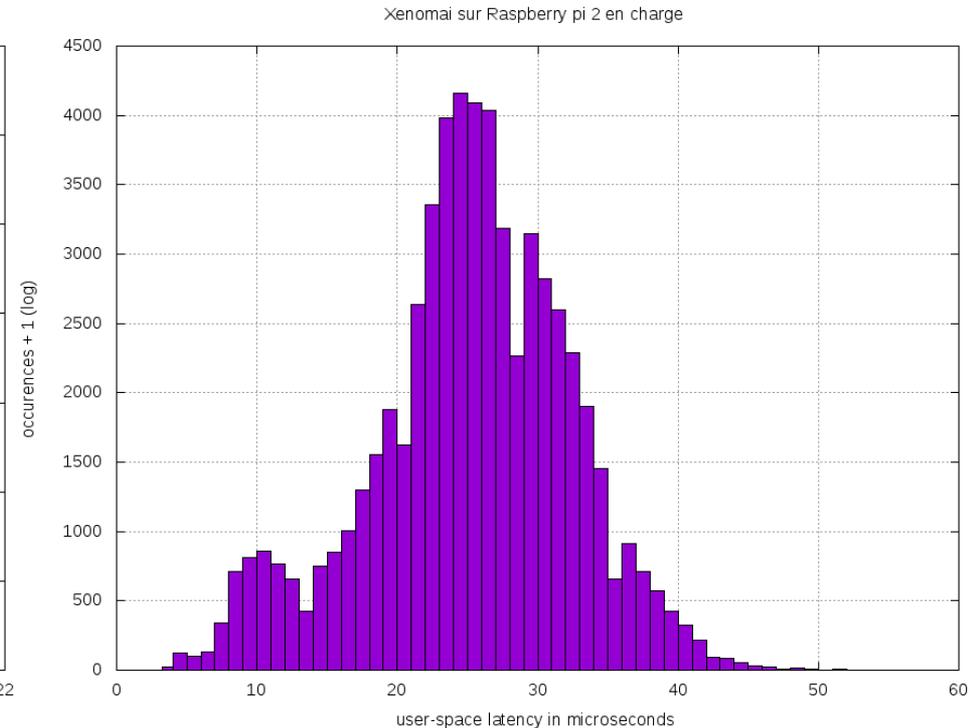
Production signal 1ms avec forte charge CPU
Linux temps réel (Xenomai)
Mesure des retards/latences

Amélioration des performances du noyau linux

Mesures des performances temps réelles sur linux



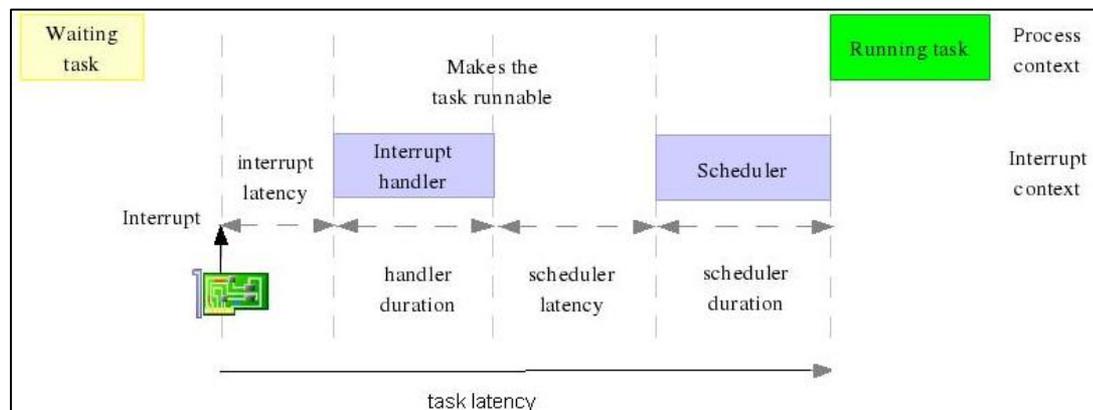
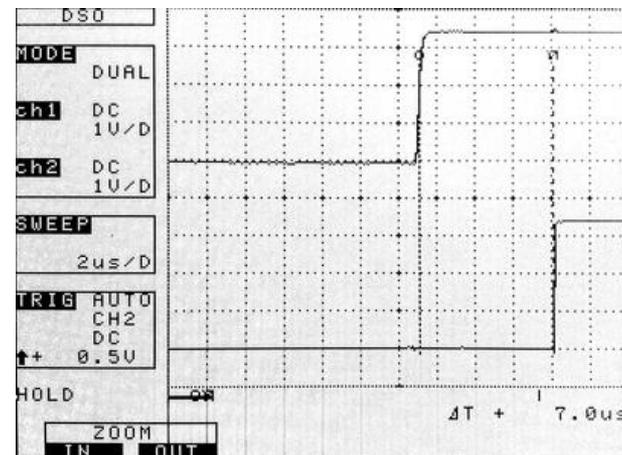
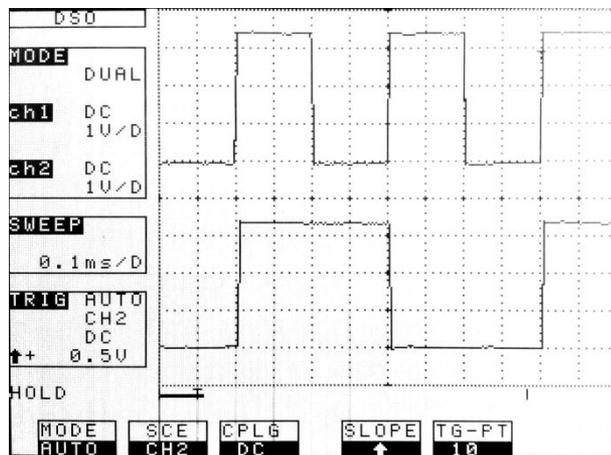
latency



stress or dohell
latency

Amélioration des performances du noyau linux

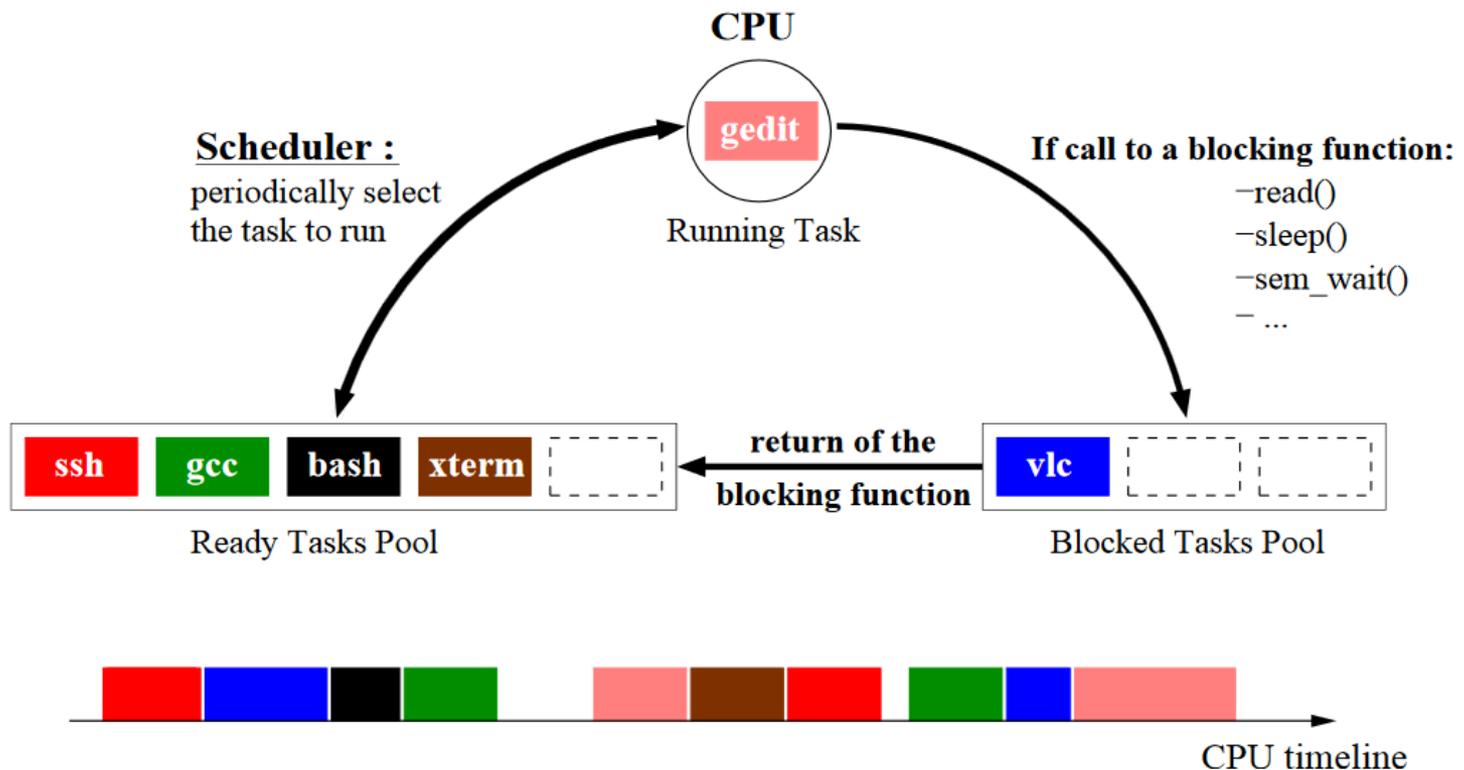
Mesures des performances temps réelles sur linux



Amélioration des performances du noyau linux

Stratégie d'ordonnancement

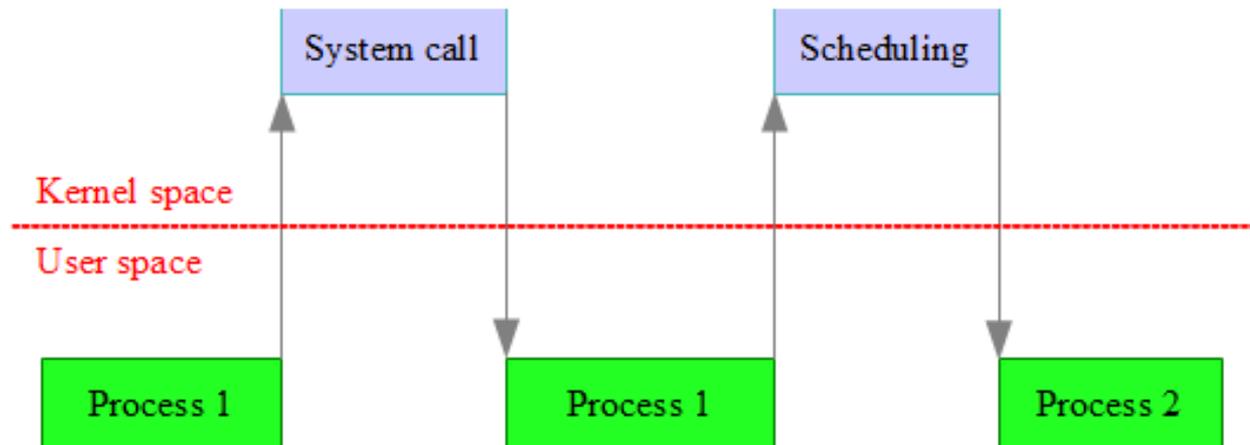
- scheduler(ordonnanceur) : sélection du processus à exécuter
- CPU partage son temps entre plusieurs processus
- 3 états pour un processus : ready, running, blocked.



Amélioration des performances du noyau linux

Stratégie d'ordonnancement

- **Préemption** : remplacement involontaire du processus s'exécutant par un autre.
- **Involontaire** : le processus en cours ne s'est pas mis en sommeil ou en attente de réception de données.



- **Noyau Linux : par défaut, pas de préemption à l'intérieur du noyau => Temps réel impossible**

Amélioration des performances du noyau linux

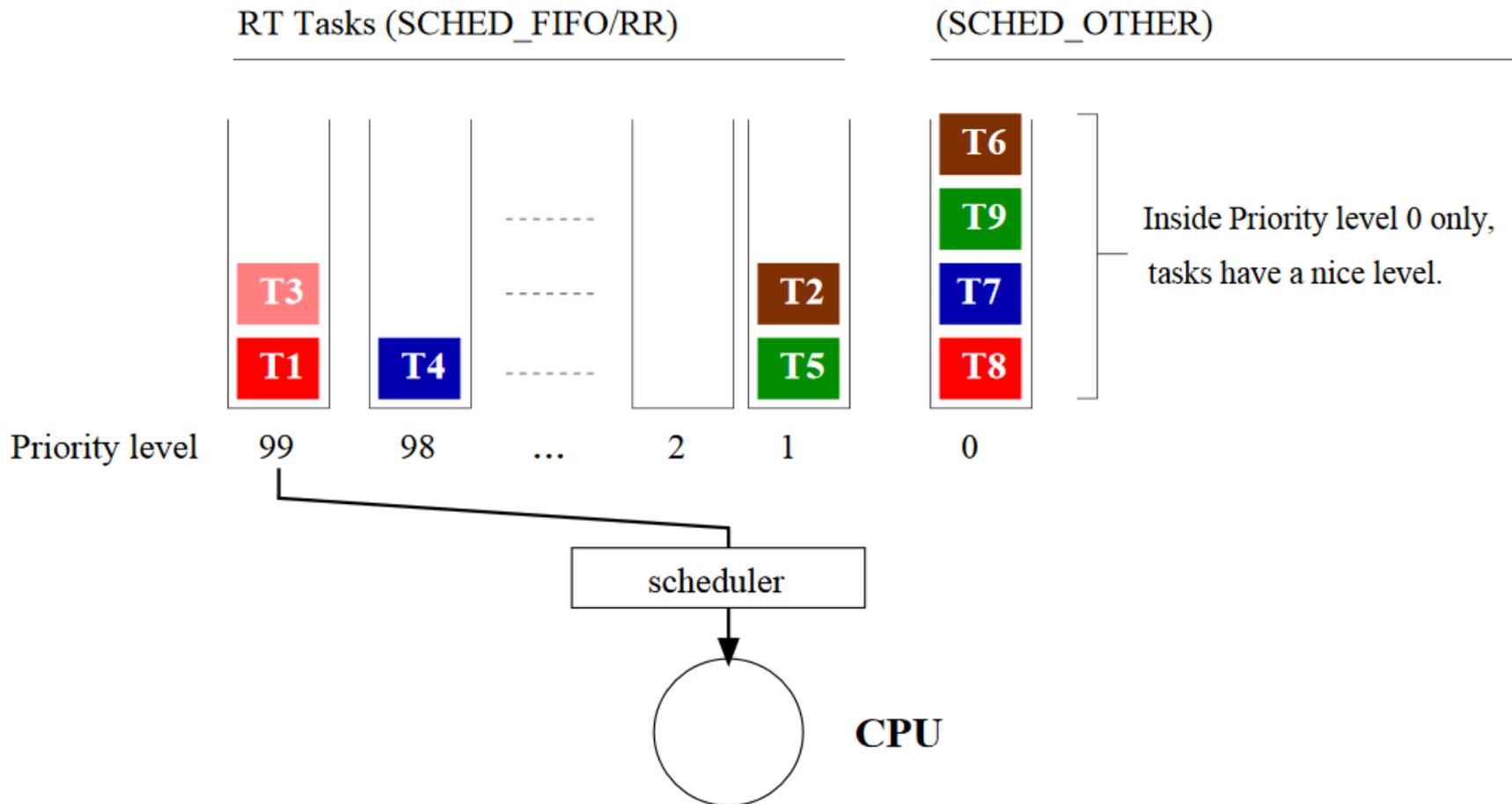
Stratégie d'ordonnancement

- **SCHED_OTHER** : ordonnanceur équitable (default)
 - favoriser des tâches : niveau de « nice »
- **SCHED_FIFO** : ordonnanceur TR à priorité fixe (need to be root user)
 - tâches utilisant cet ordonnanceur sont prioritaires sur les tâches SCHED_OTHER
 - si priorité identique, les tâches sont entièrement exécutées dans l'ordre de la file
- **SCHED_RR** : équivalent à SCHED_FIFO
 - mais une tâche interrompue est remise en bout de sa file de priorité

Amélioration des performances du noyau linux

Stratégie d'ordonnancement

Ready RT Tasks Pool



Amélioration des performances du noyau linux

Stratégie d'ordonnancement

```
#include <sched.h>

int sched_setscheduler(pid_t pid, //0: active thread
                      int policy, //SCHED_OTHER/FIFO/RR
                      const struct sched_param *p);

int sched_getscheduler(pid_t pid);

struct sched_param {
    ...
    int sched_priority; //0..99
    ...
};
```

Amélioration des performances du noyau linux

Le patch PREEMPT-RT

- **patch, élaboré par Ingo Molnar**
 - <https://wiki.linuxfoundation.org/realtime/start>
- **Principe : rendre « totalement » préemptible le code du noyau**
 - Les sections critiques (au nombre de 11 000 !)
 - Les handlers d'interruption
 - Les sections sont protégées par des spinlock ou des sémaphores et supportent l'héritage de priorité
 - Presque les 60 000 lignes : modification profonde du noyau
 - Le développement d'applications RT reposent sur l'API POSIX classique
 - Temps de latence maximum nettement amélioré

Amélioration des performances du noyau linux

Le patch PREEMPT-RT

- **Le coût de la préemption peut être important si le nombre de tâches TR augmente**
- **Temps de latence maximum nettement amélioré :**
 - Dépend largement de la plate-forme matérielle
 - Bons résultats sur x86
 - Fonctionne aussi sur ARM, Nios II, Microblaze, ...
 - dépend de la configuration logicielle
- **Disponible dans Buildroot avec le paquet `rt-tests`**
(`cyclictest`, `hackbench`, ...)

Amélioration des performances du noyau linux

Le patch PREEMPT-RT

- make linux-menuconfig

HRTIMER

```

Processor type and features
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

[*] Tickless System (Dynamic Ticks)
[*] High Resolution Timer Support
[*] Symmetric multi-processing support
  
```

```

Processor type and features
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

^(-)
(8) Maximum number of CPUs
[*] SMT (Hyperthreading) scheduler support
[*] Multi-core scheduler support
Preemption Mode (Complete Preemption (Real-Time)) --->
- *- Thread Softirqs
- *- Thread Hardirqs
  
```

IRQ = kernel thread

```

Preemption Mode
Use the arrow keys to navigate this window or press the hotkey of
the item you wish to select followed by the <SPACE BAR>. Press
<?> for additional information about this option.

( ) No Forced Preemption (Server)
( ) Voluntary Kernel Preemption (Desktop)
( ) Preemptible Kernel (Low-Latency Desktop)
[*] Complete Preemption (Real-Time)

<Select> < Help >
  
```

Linux TR sur RPi

Amélioration des performances du noyau linux

Systèmes à co-noyau - Xenomai

- **Principe : Utilisation d'un micronoyau pour satisfaire les contraintes temps réel :**
- **Le micronoyau gère :**
 - le scheduling des tâches temps réel,
 - les timers, les interruptions
 - la communication entre processus
- **Linux gère les services non temps réel**
 - connectivité réseau, USB...
 - devient une simple tâche du micronoyau

**=> Xenomai et RTAI (Real Time Application Interface)
repose sur ce principe**

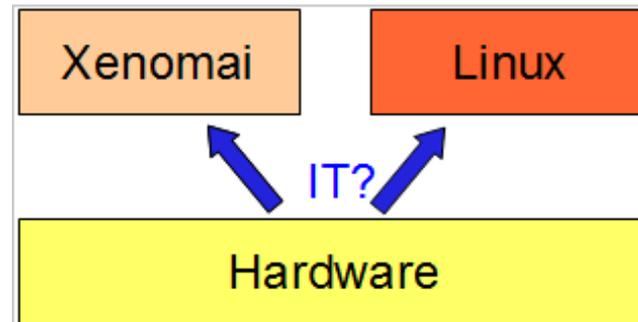
Amélioration des performances du noyau linux

Systèmes à co-noyau - Xenomai

- **Xenomai : OS temps réel qui a Linux pour tâche de fond**
- **Linux est préempté comme une simple tâche.**
 - ⇒ **La préemption impossible des sections critiques du noyau linux n'a plus d'importance**
 - ⇒ **Les tâches gérées par Xenomai apportent ainsi une garantie d'exécution temps réel dur**
- **Problème : deux OS = deux ordonnanceurs**
 - partager le matériel ?
 - faire interagir les tâches Linux et Xenomai entre-elles ?
 - traiter les interruptions ?

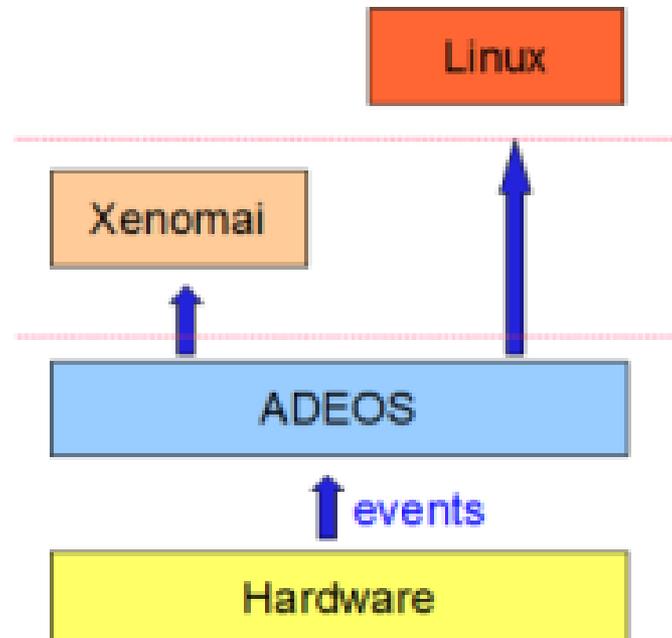
Amélioration des performances du noyau linux

Systèmes à co-noyau – Xenomai



Utilisation d'un « dispatcheur » ou « hyperviseur »

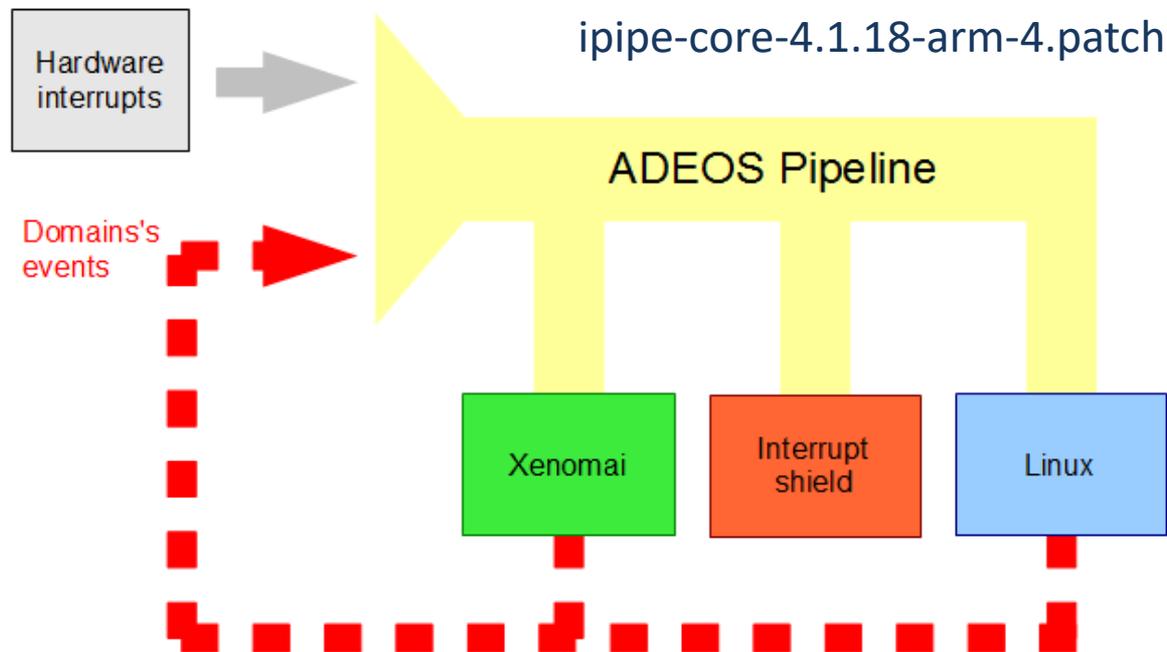
ADEOS est un nanokernel qui se présente sous la forme d'un patch du noyau Linux



Amélioration des performances du noyau linux

Systèmes à co-noyau – Xenomai

Principe du ipipe (Interrupts Pipeline) Adeos



Interrupt shield : Bloque les interruptions. Mécanisme de protection des tâches en secondary mode (scheduler de linux).
Désactivé par défaut.

Amélioration des performances du noyau linux

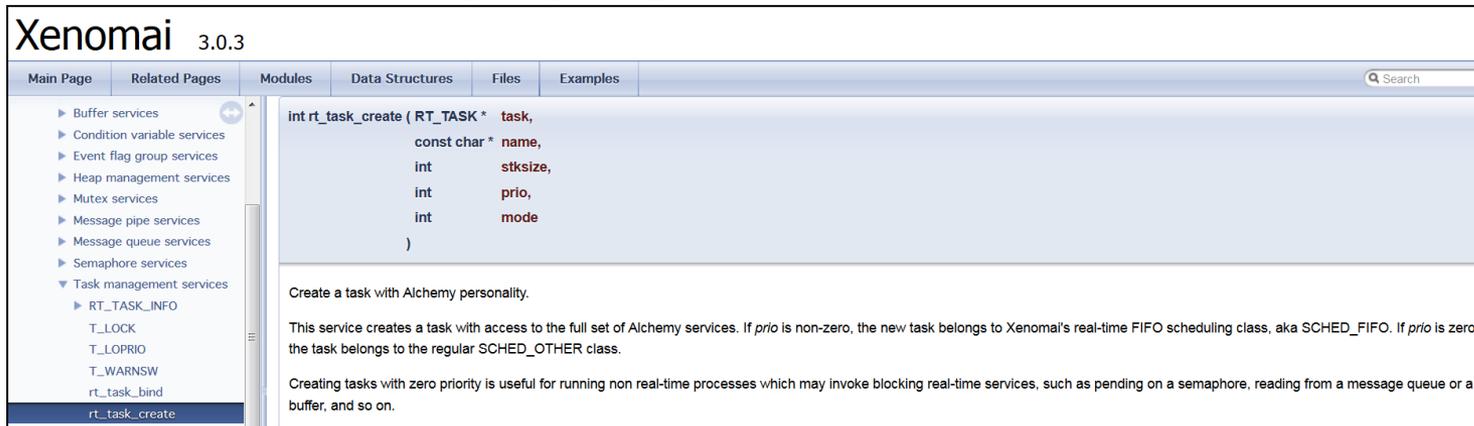
Systèmes à co-noyau – Xenomai

- **Deux modes d'exécution pour une tâches Xenomai :**
 - **Mode User (courantes)**
 - **Mode Kernel**
- **Un grand nombre d'architectures supportées :**
 - <http://xenomai.org/embedded-hardware/>
- **API et skins :**
 - **API (Native) très complète :**
 - tâches avec 99 niveaux de priorité, round robin optionnel...
 - files de messages ;
 - allocation dynamique de mémoire spécifique RT ;
 - sémaphores ;
 - watchdogs ;
 - timers ;
 - mutexes ;
 - ...

Amélioration des performances du noyau linux

Systèmes à co-noyau – Xenomai

- API et skins :
- **Migrating from Xenomai 2.x to 3.x**
 - Nouvelle API : **alchemy** remplace **native**
 - <http://xenomai.org/migrating-from-xenomai-2-x-to-3-x/>



The screenshot shows the Xenomai 3.0.3 website. The main content area displays the function signature for `rt_task_create` and a description of the service. The function signature is:

```
int rt_task_create ( RT_TASK * task,
                   const char * name,
                   int stksize,
                   int prio,
                   int mode
                   )
```

Below the signature, the text reads: "Create a task with Alchemy personality." and "This service creates a task with access to the full set of Alchemy services. If *prio* is non-zero, the new task belongs to Xenomai's real-time FIFO scheduling class, aka SCHED_FIFO. If *prio* is zero, the task belongs to the regular SCHED_OTHER class." A note at the bottom states: "Creating tasks with zero priority is useful for running non real-time processes which may invoke blocking real-time services, such as pending on a semaphore, reading from a message queue or a buffer, and so on."

- Les skins sont des API d'autres RTOS (pSos, VxWorks...) qui encapsulent des appels natifs
- Prototypage
- Portage d'applications existantes

- **Abraham Silberschatz, Peter Baer Galvin, Greg Gagne :**
Operating System Concepts (2011)
- **Andrew Tanenbaum :** Systèmes d'exploitation (3ème édition, 2008)
- **Gilles Blanc :** Linux embarqué (2011)
- **Pierre Ficheux :** Linux embarqué : (3è édition, 2010)
- **Cours de Stéphane Huet :** Principes des OS / Linux embarqué (2014)
- **Christophe Blaess :** Ingéniérie et formations sur les systèmes libres
<http://www.blaess.fr/christophe/>
- **Cours de Jalil Boukhobza :** Systèmes d'exploitation pour l'embarqué
<http://syst.univ-brest.fr/~boukhobza/index.php/systemes-dexploitation-pour-lembarque>
- **Cours de Hugo Descoubes :** Architecture des ordinateurs
https://www.canal-u.tv/producteurs/centre_d_enseignement_multimedia_universitaire_c_e_m_u/ensicaen/architecture_et_technologie_des_ordinateurs
- **Free electron :** Formation Buildroot
<http://free-electrons.com/doc/training/buildroot/buildroot-slides.pdf>
- **Tutoriel Premiers pas avec Xenomai :** David CHABAL
<http://dchabal.developpez.com/tutoriels/linux/xenomai/>