Université Polytechnique Hauts-de-France

INSA Hauts-de-France

Année 2024 - 2025

5A ESE - O9MA332

Architecture Temps Réel Pour L'embarqué

TD n°1 : Installation d'un noyau temps réel Xenomai sur Raspberry Pi4

Marwane AYAIDA (marwane.ayaida@uphf.fr)

1 Objectifs et pré-requis du TD

1.1 Objectifs

L'objectif de ce TD est de vous aider à installer un noyau Linux que vous allez configurer vous-même. Par la suite, vous allez patcher le noyau Linux pour installer un co-noyau temps-réel Xenomai.
La première chose à faire est de bien choisir la version du Linux et du patch Xenomai qui va avec. Pour ce tutorial, j'ai choisi après plusieurs tests la version Linux "*Linux raspberrypi 4.19.127*" et la version Xenomai qui va avec "*3.1*" et I-Pipe qui va avec aussi "*4.19.82*".
Il est à noter aussi que j'ai testé ce tutorial sur une machine virtuelle ***Ubuntu 20.04.6*** avec *GCC* version *gcc-arm-linux-gnueabihf* (4 :9.3.0-1ubuntu2). De plus, le noyau compilé a été testé sur un Raspberry Pi3 Model B et un Raspberry Pi3 Model B+ ainsi qu'un Raspberry Pi4.
Enfin, il se pourra que vous devriez re-compiler votre noyau pour rajouter des options utiles pour votre projet. Si cela marche pour vous une fois, vous pourrez après le faire autant de fois qu'il est nécessaire en modifiant les options du noyau.

1.2 Préparation du Raspberry Pi4

Le premier pré-requis de ce TP est d'avoir un Raspberry Pi4 (cela devrait aussi fonctionner pour les modèles Pi3 et Pi3 B+ et même Pi3 Model B).
Au début, nous avons besoin d'un système d'exploitation ***Raspbian*** qui tourne sur le Raspberry. Pour cela, vous avez besoin de l'outil ***Raspberry Pi Imager*** pour installer l'OS sur la carte SD. Ainsi, vous pourrez suivre les étapes suivantes l'installer :

1. Connectez-vous sur le site Web : https ://www.raspberrypi.org/downloads/
2. Sélectionnez l'outil qui correspond à votre système d'exploitation.
3. Ouvrez l'installeur téléchargé et éxecutez-le.

Une fois l'outil installé, nous allons télécharger un OS ***Raspbian*** spécifique pour éviter tout problème de compilation par la suite. Suivez les étapes suivantes :

1. Téléchargez le fichier : "*2021-03-04-raspios-buster-armhf.zip*" depuis ce lien : https ://downloads.raspberrypi.com/raspios_armhf/images/raspios_armhf-2021-03-25/2021-03-04-raspios-buster-armhf.zip
2. Décompressez le fichier téléchargé pour obtenir le fichier "*2021-03-04-raspios-buster-armhf.img*".

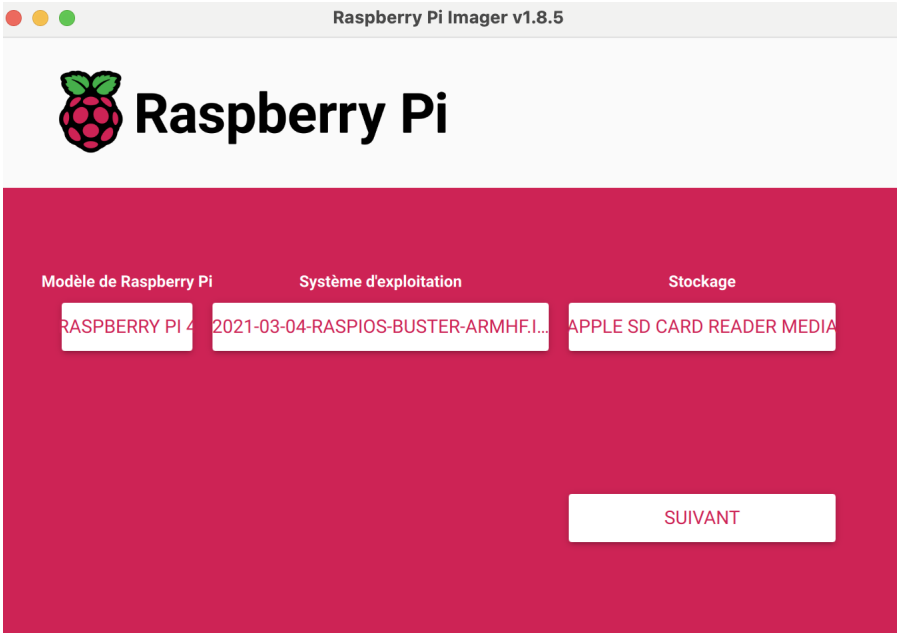

FIGURE 1 – Raspberry Pi Imager selecting OS and SD Card

3. Maintenant, démarrez l'outil : ***Raspberry Pi Imager***.
4. Sélectionnez le modèle Raspberry Pi4.
5. Choisissez l'OS ***Raspbian*** "*2021-03-04-raspios-buster-armhf.img*" en cliquant sur "*Use custom*" comme le montre la figure 1.
6. Sélectionnez l'emplacement de la carte SD.
7. Cliquez sur "*SUIVANT*".
8. Cliquez sur "*MODIFIER RÉGLAGES*".
9. Dans l'Onglet "*GÉNÉRAL*", choisisez un Nom d'ultilisateur et un Mot de passe, ainsi que les réglages locaux comme affiché sur la figure 2.
10. Dans l'Onglet "*SERVICES*", activez le service "*SSH*" comme représenté par la figure 3.
11. Cliquez sur "*ENREGISTRER*".
12. Lancez l'installation en cliquant sur *OUI* et puis encore *OUI*.
13. Enfin, attendez la fin du formatage et de l'installation qui pourra prendre un certain moment.

Ainsi, vous obtenez un système ***Raspbian*** fonctionnel. Pour se connecter au Raspberry en Ethernet depuis son ordinateur, il faut d'abord attribuer une adresse IP fixe au Raspberry. Pour ce faire, récupérez la carte SD du Raspberry et lisez la avec l'ex-

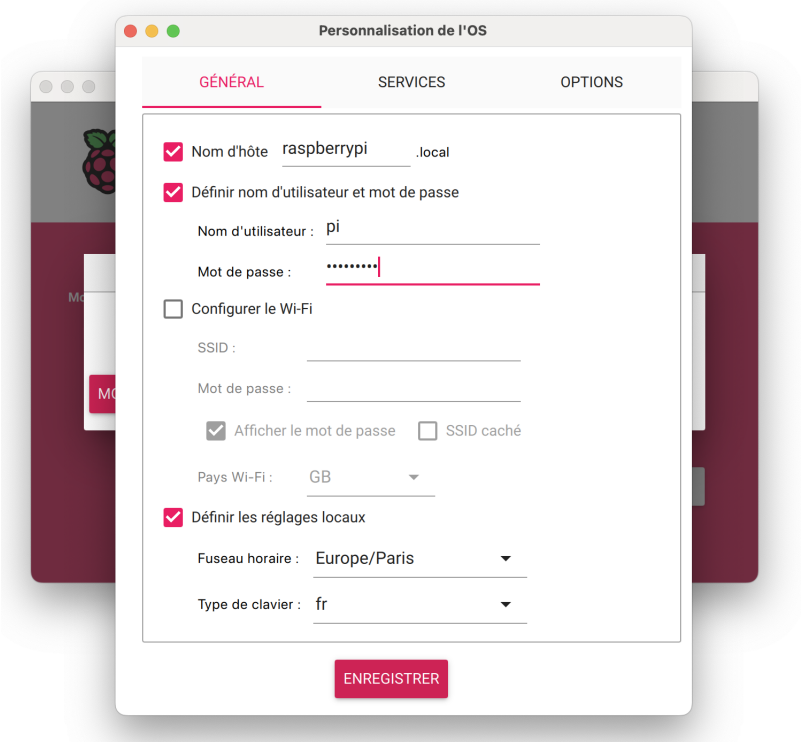

FIGURE 2 – Raspberry Pi Imager configuration of the OS

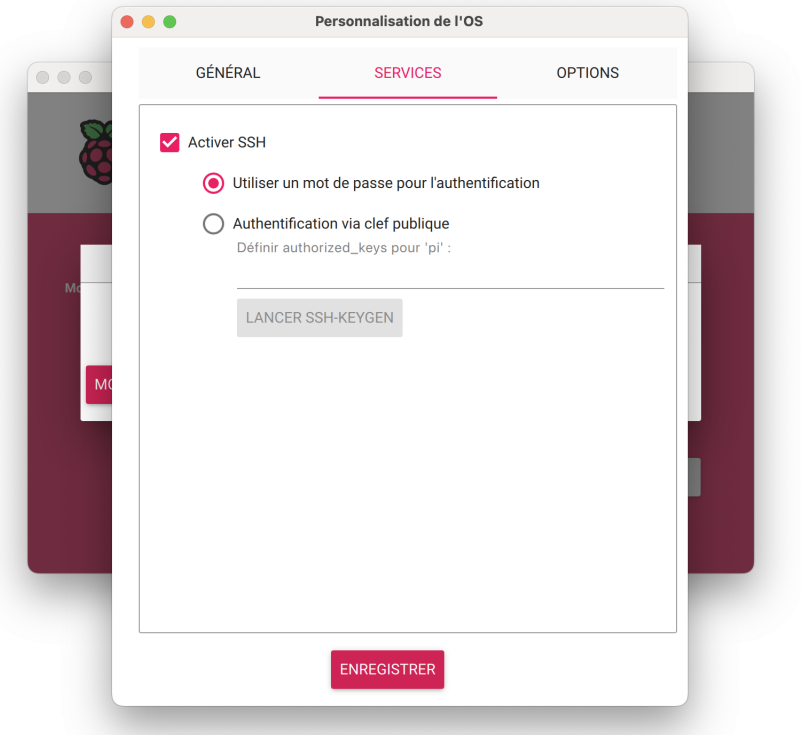

FIGURE 3 – Raspberry Pi Imager configuration of the OS

plorateur de fichiers de Windows, Linux ou de MAC (la carte SD est normalement appelée "***boot***").
Cherchez le fichier "*cmdline.txt* ", c'est ce fichier qu'on va modifier. Vous pouvez faire une sauvegarde du fichier avant de le modifier pour plus de sécurité. Ouvrez donc le fichier avec votre éditeur de texte favori et ajoutez à la fin de la ligne (pas de saut de ligne) le texte suivant : *ip=192.168.0.X* (X à choisir entre 2 et 253), par exemple : *192.168.0.100*. L'adresse se terminant par 1 est réservée à votre PC : *192.168.0.1*.
Notez l'IP choisie quelque part. Une fois terminé, enregistrez le fichier (Ctrl+S) et retirez la carte. Ré-insérez la dans le Raspberry Pi.
Maintenant, on peut brancher le Raspberry au PC/Mac à l'aide des ports Ethernet de chacun à l'aide d'un câble RJ45 et alimenter le Raspberry au 5V. On attend 1-2 minutes le temps que le Raspberry s'initialise.
Il ne faut pas oublier de mettre votre PC/MAC dans le même sous-réseau pour permettre la connexion.
Une fois le Raspberry est initialisé, on peut vérifier que tout marche en ouvrant l'invite de commande Windows ou le Terminal sur MAC/Linux en tapant (en suivant l'exemple utilisé plus haut) :

```
1 ~$ ping <ipaddress>
```

Il faudra penser à remplacer *<ipaddress>* par l'IP de votre Raspberry Pi. S'il n'y a pas d'erreur, le Raspberry est prêt à être connecté en ***SSH*** par Ethernet.
Maintenant que vous êtes capable de vous connecter en ***SSH*** à partir de votre PC/MAC, vous pourrez utiliser les outils ***Putty***, ***Tera Term*** ou juste le ***Terminal*** Linux ou MAC pour vous y connecter, par exemple en tapant dans un ***Terminal*** :

```
1 ~$ ssh pi@<ipaddress>
```

De même, il faudra penser à remplacer *<ipaddress>* par l'IP de votre Raspberry Pi. Puis, saisissez le mot de passe renseigné précédemment.
On peut ainsi savoir la version de Linux installé :

```
1 ~$ uname -a
2 Linux raspberrypi 5.10.17-v7l+ #1403 SMP Mon Feb 22 11:33:35 GMT
3 2021 armv7l GNU/Linux
```

Cela nous permettra de comparer avec le système temps-réel que l'on installera par la suite.

2 Téléchargement des ressources Linux

Puisque vous compilerez sur un PC hôte à la place de le faire directement sur le Raspberry afin de gagner du temps, il faudra alors que vous revenez sur votre PC hôte afin de commencer la cross-compilation.

2.1 Préparation de l'installation

Pour démarrer, créez un nouveau dossier afin d'avoir une installation propre :

```
1 ~$ mkdir rpi-kernel
2
3 ~$ cd rpi-kernel
4
5 ~/rpi-kernel$ mkdir rt-kernel
```

Le dossier ***rt-kernel*** sera utilisé pour collecter les fichiers compilés.

2.2 Obtention des codes sources

Pour obtenir les codes sources, vous utiliserez la commande ***GIT***. Si ce n'est pas déjà fait sur votre PC, commencez par l'installer :

```
1 ~/rpi-kernel$ sudo apt update
2 ...
3 Lecture des listes de paquets... Fait
4 Construction de l'arbre des dependances... Fait
5 Lecture des informations d'etat... Fait
6 Tous les paquets sont a jour.
7
8 ~/rpi-kernel$ sudo apt install git
9 ...
10 Depaquetage de git (1:2.34.1-1ubuntu1.10) ...
11 Parametrage de liberror-perl (0.17029-1) ...
12 Parametrage de git-man (1:2.34.1-1ubuntu1.10) ...
13 Parametrage de git (1:2.34.1-1ubuntu1.10) ...
14 Traitement des actions differees (" triggers ") pour man-db (2.10.2-1) ...
```

Installez aussi les outils nécessaires pour la compilation du noyau :

```
1 ~/rpi-kernel$ sudo apt install ncurses-dev bc build-essential bison flex
2           pkg-config qtbase5-dev libssl-dev autoconf libtool
3 [sudo] Mot de passe de marwane :
```

```
Lecture des listes de paquets... Fait
Construction de l arbre des dependances
Lecture des informations d etat... Fait
...
Parametrage de libx11-dev:amd64 (2:1.6.9-2ubuntu1.6) ...
Parametrage de libxext-dev:amd64 (2:1.3.4-0ubuntu1) ...
Parametrage de libglx-dev:amd64 (1.3.2-1~ubuntu0.20.04.2) ...
Parametrage de libgl-dev:amd64 (1.3.2-1~ubuntu0.20.04.2) ...
Parametrage de libegl-dev:amd64 (1.3.2-1~ubuntu0.20.04.2) ...
Parametrage de libglu1-mesa-dev:amd64 (9.0.1-1build1) ...
Parametrage de qtbase5-dev:amd64 (5.12.8+dfsg-0ubuntu2.1) ...
Parametrage de libqt5opengl5-dev:amd64 (5.12.8+dfsg-0ubuntu2.1) ...
```

Clonez les dossiers Linux Version 4.19.127 :

```
        ~/rpi-kernel$ git clone -b rpi-4.19.y
        https://github.com/raspberrypi/linux.git linux-rpi-4.19.127-xeno3
        Clonage dans 'linux-rpi-4.19.86-xeno3'...
        remote: Enumerating objects: 11942637, done.
        remote: Counting objects: 100% (6/6), done.
        remote: Compressing objects: 100% (4/4), done.
        remote: Total 11942637 (delta 3), reused 2 (delta 2),
                pack-reused 11942631 (from 1)
        Reception d'objets: 100% (11942637/11942637),
                3.97 Gio | 22.19 Mio/s, fait.
        Resolution des deltas: 100% (10000213/10000213), fait.
        Verification des objets: 100% (33554432/33554432), fait.
        Mise a jour des fichiers: 100% (62377/62377), fait.
```

Créez un dossier lié pour un accès facile :

```
        ~/rpi-kernel$ ln -s linux-rpi-4.19.127-xeno3 linux
```

Le code source du noyau Linux sera téléchargé dans le dossier ***linux-rpi-4.19.127-xeno3*** pour une taille de 1 à 2 Go. Cela peut prendre un certain temps selon votre débit de connexion.

Vous avez également besoin des codes sources du noyau Xenomai et du patch correspondant. Vous devez choisir la version stable v3.1 et ainsi le patch pour arm qui va avec :

```
~/rpi-kernel$ wget https://ftp.denx.de/pub/xenomai/ipipe/v4.x/arm/ipipe-
                core-4.19.82-arm-6.patch
--2024-09-17 12:32:48-- https://ftp.denx.de/pub/xenomai/ipipe/
```

```
          v4.x/arm/ipipe-core-4.19.82-arm-6.patch
Resolution de ftp.denx.de (ftp.denx.de)... 85.214.49.3, 2a01:238:43f4:4600:bc64:4f44:381f:1163
Connexion a ftp.denx.de (ftp.denx.de)|85.214.49.3|:443... connecte.
requete HTTP transmise, en attente de la reponse... 200 OK
Taille : 611977 (598K) [application/octet-stream]
Enregistre : "ipipe-core-4.19.82-arm-6.patch"

ipipe-core-4.19.82-arm-6.patch 100%[==================
==========================================
==============>] 597,63K --.-KB/s ds 0,1s


2023-10-14 23:10:13 (4,37 MB/s) - 'ipipe-core-4.19.82-arm-6.patch'
           enregistre [611977/611977]

~/rpi-kernel$ wget https://ftp.denx.de/pub/xenomai/xenomai/stable/
                    xenomai-3.1.tar.bz2
--2024-09-17 12:36:07-- https://ftp.denx.de/pub/xenomai/xenomai/
          stable/xenomai-3.1.tar.bz2
Resolution de ftp.denx.de (ftp.denx.de)... 85.214.49.3,
          2a01:238:43f4:4600:bc64:4f44:381f:1163
Connexion a ftp.denx.de (ftp.denx.de)|85.214.49.3|:443... connecte.
requete HTTP transmise, en attente de la reponse... 200 OK
Taille : 2498379 (2,4M) [application/octet-stream]
Enregistre : "xenomai-3.1.tar.bz2"

xenomai-3.1.tar.bz2 100%[=========================
==========================================
======>] 2,38M 12,6MB/s ds 0,2s

2024-09-17 12:36:07 (12,6 MB/s) - "xenomai-3.1.tar.bz2"
          enregistre [2498379/2498379]


2023-10-14 23:16:38 (10,9 MB/s) - 'xenomai-3.1.tar.bz2'
                    enregistre [2498379/2498379]

~/rpi-kernel$ tar -xjvf xenomai-3.1.tar.bz2
xenomai-3.1/
```

```
xenomai-3.1/include/
xenomai-3.1/include/mercury/
xenomai-3.1/include/mercury/Makefile.am
xenomai-3.1/include/mercury/boilerplate/
xenomai-3.1/include/mercury/boilerplate/Makefile.am
xenomai-3.1/include/mercury/boilerplate/sched.h
xenomai-3.1/include/mercury/boilerplate/wrappers.h
xenomai-3.1/include/mercury/boilerplate/Makefile.in
xenomai-3.1/include/mercury/boilerplate/trace.h
xenomai-3.1/include/mercury/boilerplate/limits.h
xenomai-3.1/include/mercury/boilerplate/signal.h
xenomai-3.1/include/mercury/Makefile.in
xenomai-3.1/include/mercury/pthread.h
...
xenomai-3.1/demo/posix/cobalt/
xenomai-3.1/demo/posix/cobalt/can-rtt.c
xenomai-3.1/demo/posix/cobalt/gpiopwm.c
xenomai-3.1/demo/posix/cobalt/eth_p_all.c
xenomai-3.1/demo/posix/cobalt/Makefile.am
xenomai-3.1/demo/posix/cobalt/bufp-readwrite.c
xenomai-3.1/demo/posix/cobalt/bufp-label.c
xenomai-3.1/demo/posix/cobalt/Makefile.in
xenomai-3.1/demo/posix/cobalt/xddp-label.c
xenomai-3.1/demo/posix/cobalt/xddp-stream.c
xenomai-3.1/demo/posix/cobalt/iddp-label.c
xenomai-3.1/demo/posix/cobalt/iddp-sendrecv.c
xenomai-3.1/demo/posix/cobalt/xddp-echo.c
```

Pensez à sauvegarder le code source de Linux car on en aura besoin par la suite lors du TP3 pour le mettre sur le Raspberry Pi :

```
~/rpi-kernel/linux$ cd linux/
~/rpi-kernel/linux$ tar czf ../linux-src.tgz *
```

Ce patch présente un souci avec deux fichiers lors de l'application de ce patch qui sont ***irq-bcm2835.c*** et ***irq-bcm2836.c***.
Pour cela, il faudra télécharger ces deux fichiers qui corrigent ce problème :

```
~/rpi-kernel/linux$ cd ..

~/rpi-kernel$ wget https://raw.githubusercontent.com/cpb-/xeno-pi/
        7169258f33fab2b1823a75593130189efbb55979/add-arm-8-a-
```

```
          architecture-to-xenomai-3.1.patch
--2024-09-17 12:54:27-- https://raw.githubusercontent.com/cpb-/
xeno-pi/7169258f33fab2b1823a75593130189efbb55979/
add-arm-8-a-architecture-to-xenomai-3.1.patch
Resolution de raw.githubusercontent.com (raw.githubusercontent.com)...
 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connexion a raw.githubusercontent.com (raw.githubusercontent.com)|
185.199.110.133|:443... connecte.
requete HTTP transmise, en attente de la reponse... 200 OK
Taille : 541 [text/plain]
Enregistre : 'add-arm-8-a-architecture-to-xenomai-3.1.patch'

add-arm-8-a-architecture-to-xenoma 100%
[==========================================
==========================>] 541 --.-KB/s ds 0s

2024-09-17 12:54:27 (10,7 MB/s) - 'add-arm-8-a-architecture-to-
xenomai-3.1.patch' enregistre [541/541]

~/rpi-kernel$ wget https://raw.githubusercontent.com/cpb-/xeno-pi/
          7169258f33fab2b1823a75593130189efbb55979/pre-ipipe-
          core-4.19.82-arm-6.patch
--2024-09-17 12:57:06-- https://raw.githubusercontent.com/cpb-/
xeno-pi/7169258f33fab2b1823a75593130189efbb55979/
pre-ipipe-core-4.19.82-arm-6.patch
Resolution de raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connexion a raw.githubusercontent.com (raw.githubusercontent.com)|
185.199.110.133|:443... connecte.
requete HTTP transmise, en attente de la reponse... 200 OK
Taille : 1485 (1,5K) [text/plain]
Enregistre : 'pre-ipipe-core-4.19.82-arm-6.patch'

pre-ipipe-core-4.19.82-arm-6.pat 100%
[==========================================
==================>] 1,45K --.-KB/s ds 0s

2024-09-17 12:57:07 (10,6 MB/s) - 'pre-ipipe-core-4.19.82-arm-
6.patch' enregistre [1485/1485]
```

```
44
45 ~/rpi-kernel$ wget https://raw.githubusercontent.com/cpb-/xeno-pi/
46           7169258f33fab2b1823a75593130189efbb55979/post-ipipe-
47           core-4.19.82-arm-6.patch
48 --2024-09-17 12:58:50-- https://raw.githubusercontent.com/cpb-/
49 xeno-pi/7169258f33fab2b1823a75593130189efbb55979/post-ipipe-core-
50 4.19.82-arm-6.patch
51 Resolution de raw.githubusercontent.com (raw.githubusercontent.com)...
52 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
53 Connexion a raw.githubusercontent.com (raw.githubusercontent.com)|
54 185.199.110.133|:443... connecte.
55 requete HTTP transmise, en attente de la reponse... 200 OK
56 Taille : 1283 (1,3K) [text/plain]
57 Enregistre : 'post-ipipe-core-4.19.82-arm-6.patch'
58
59 post-ipipe-core-4.19.82-arm-6.pa 100%
60 [=======================================
61 ==================>] 1,25K --.-KB/s ds 0s
62
63 2024-09-17 12:58:50 (24,6 MB/s) - 'post-ipipe-core-4.19.82-arm-
64 6.patch' enregistre [1283/1283]
```

Par la suite, appliquez le patch pour résoudre le problème dans le code source du noyau Linux :

```
1 ~/rpi-kernel$ cd xenomai-3.1/
2
3 ~/rpi-kernel/xenomai-3.1$ patch -p1 < ../add-arm-8-a-
4                     architecture-to-xenomai-3.1.patch --verbose
5 Hmm... Looks like a unified diff to me...
6 The text leading up to this was:
7 --------------------------
8 |diff -ruN a/lib/cobalt/arch/arm/include/asm/xenomai/features.h
9           b/lib/cobalt/arch/arm/include/asm/xenomai/features.h
10 |--- a/lib/cobalt/arch/arm/include/asm/xenomai/features.h
11           2018-10-02 17:39:36.000000000 +0200
12 |+++ b/lib/cobalt/arch/arm/include/asm/xenomai/features.h
13           2020-02-25 12:28:10.225480377 +0100
14 --------------------------
15 patching file lib/cobalt/arch/arm/include/asm/xenomai/features.h
16 Using Plan A...
```

```
Hunk #1 succeeded at 48.
Done


~/rpi-kernel/xenomai-3.1$ cd ../linux

~/rpi-kernel/linux$ patch -p1 < ../pre-ipipe-core-4.19.82-
        arm-6.patch --verbose
Hmm... Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|diff --git a/drivers/irqchip/irq-bcm2835.c
        b/drivers/irqchip/irq-bcm2835.c
|index 72abca2bac3a..2c8c8ad710e3 100644
|--- a/drivers/irqchip/irq-bcm2835.c
|+++ b/drivers/irqchip/irq-bcm2835.c
--------------------------
patching file drivers/irqchip/irq-bcm2835.c
Using Plan A...
Hunk #1 succeeded at 172 (offset -1 lines).
...
--------------------------
patching file lib/dump_stack.c
Using Plan A...
Hunk #1 succeeded at 106.
Done


~/rpi-kernel/linux$ patch -p1 < ../ipipe-core-4.19.82-
        arm-6.patch --verbose
Hmm... Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|diff --git a/Documentation/ipipe-arm.rst
        b/Documentation/ipipe-arm.rst
|new file mode 100644
|index 000000000000..71ba475543c3
|--- /dev/null
|+++ b/Documentation/ipipe-arm.rst
```

```
56 ----------------------------
57 patching file Documentation/ipipe-arm.rst
58 Using Plan A...
59 Hunk #1 succeeded at 1.
60 ...
61 patching file mm/mprotect.c
62 Using Plan A...
63 Hunk #1 succeeded at 22.
64 Hunk #2 succeeded at 43.
65 Hunk #3 succeeded at 111.
66 Hunk #4 succeeded at 124.
67 Hunk #5 succeeded at 339 (offset 34 lines).
68 Hmm... The next patch looks like a unified diff to me...
69 The text leading up to this was:
70 ----------------------------
71 |diff --git a/mm/vmalloc.c b/mm/vmalloc.c
72 |index d8e877365f9f..eb30c1f87056 100644
73 |--- a/mm/vmalloc.c
74 |+++ b/mm/vmalloc.c
75 ----------------------------
76 patching file mm/vmalloc.c
77 Using Plan A...
78 Hunk #1 succeeded at 233 (offset 1 line).
79 done
80 Removing file drivers/irqchip/irq-atmel-aic.c
81
82
83 ~/rpi-kernel/linux$ patch -p1 < ../post-ipipe-core-4.19.82-
84           arm-6.patch --verbose
85 Hmm... Looks like a unified diff to me...
86 The text leading up to this was:
87 ----------------------------
88 |diff --git a/drivers/irqchip/irq-bcm2835.c
89           b/drivers/irqchip/irq-bcm2835.c
90 |index d49596f23618..d0e450a4626c 100644
91 |--- a/drivers/irqchip/irq-bcm2835.c
92 |+++ b/drivers/irqchip/irq-bcm2835.c
93 ----------------------------
94 patching file drivers/irqchip/irq-bcm2835.c
```

```
Using Plan A...
Hunk #1 succeeded at 173 (offset -1 lines).
Hmm... The next patch looks like a unified diff to me...
The text leading up to this was:
--------------------------
|diff --git a/drivers/irqchip/irq-bcm2836.c
|        b/drivers/irqchip/irq-bcm2836.c
|index a5ae07b6af3f..9d1e8da44c09 100644
|--- a/drivers/irqchip/irq-bcm2836.c
|+++ b/drivers/irqchip/irq-bcm2836.c
--------------------------
patching file drivers/irqchip/irq-bcm2836.c
Using Plan A...
Hunk #1 succeeded at 201.
Hmm... The next patch looks like a unified diff to me...
The text leading up to this was:
--------------------------
|diff --git a/lib/dump_stack.c b/lib/dump_stack.c
|index 1b734cb90f79..db1766841395 100644
|--- a/lib/dump_stack.c
|+++ b/lib/dump_stack.c
--------------------------
patching file lib/dump_stack.c
Using Plan A...
Hunk #1 succeeded at 133.
Done
```

2.3 Préparation de l'environnement pour la compilation

Il faut patcher le noyau pour rajouter ***iPipe*** d'***ADEOS*** et ***Xenomai*** dans le domaine primaire et ***Linux*** dans le domaine secondaire :

```
~/rpi-kernel/linux$ cd ..
~/rpi-kernel$ xenomai-3.1/scripts/prepare-kernel.sh --linux=linux/
        --arch=arm --ipipe=ipipe-core-4.19.82-arm-6.patch --verbose
Preparing kernel 4.19.127 in /home/marwane/rpi-kernel/linux...
I-pipe found - bypassing patch.
I-pipe core/arm #6 installed.
Links installed.
Build system ready.
```

Enfin, il faut préparer les variables de l'environnement pour la compilation :

```
~/rpi-kernel$ export ARCH=arm

~/rpi-kernel$ export CROSS_COMPILE=arm-linux-gnueabihf-

~/rpi-kernel$ export INSTALL_MOD_PATH=~/rpi-kernel/rt-kernel

~/rpi-kernel$ export INSTALL_DTBS_PATH=~/rpi-kernel/rt-kernel
```

Pour plus de détails sur les différentes variables :

- ARCH : l'architecture du système (ici "***arm***" pour le Raspberry).
- CROSS_COMPILE : le chemin vers l'outil de la chaîne de cross-compilation.
- INSTALL_MOD_PATH : le chemin vers lequel les modules du noyau compilés (MOD) vont être installés.
- INSTALL_DTBS_PATH : le chemin vers lequel les fichiers compilés DTB (Device Tree Blob) vont être installés.

Notez bien que l'export des variables est valable juste dans le Terminal que vous utilisez. Donc, si vous fermez ce dernier ou vous ouvrez un deuxième, il faudra refaire l'export des variables. Si vous souhaitez vérifier si l'export des variables est bien fait :

```
~/rpi-kernel$ echo $ARCH
arm
```

Si cela ne donne rien, c'est qu'il faudra refaire l'export des variables.

3 Compilation du noyau

3.1 Configuration du noyau

Pour établir la configuration du noyau, tapez les commandes qui suivent :

```
~/rpi-kernel$ export KERNEL=kernel7l

~/rpi-kernel$ cd linux/

~/rpi-kernel/linux/$ make bcm2711_defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
YACC scripts/kconfig/zconf.tab.c
LEX scripts/kconfig/zconf.lex.c
HOSTCC scripts/kconfig/zconf.tab.o
```

```
11 HOSTLD scripts/kconfig/conf
12 #
13 # configuration written to .config
```

Il est à remarquer ici que "***bcm2711_defconfig***" est utilisé pour Raspberry Pi 4. Par contre pour Raspberry Pi 2, 3 et 3 B(+), il faut plutôt utiliser "***bcm2709_defconfig***". L'étape de configuration du noyau peut alors commencer :

```
1 ~/rpi-kernel/linux/$ make xconfig
2   UPD scripts/kconfig/.qconf-cfg
3 MOC scripts/kconfig/qconf.moc
4 HOSTCXX scripts/kconfig/qconf.o
5 HOSTLD scripts/kconfig/qconf
6 scripts/kconfig/qconf Kconfig
7 #
8 # configuration written to .config
9 #
```

Il existe environ 15 000 options de configuration pour le noyau Linux version 5 ! C'est cette étape qui est la plus importante et la plus fastidieuse. Il peut y avoir plusieurs tests de configurations différents afin d'obtenir la configuration optimale pour votre système. De plus, c'est surtout cette étape qui vous permettra d'obtenir un système d'exploitation light embarqué en ne cochant que les modules dont vous avez besoin et qui sont utiles pour votre plateforme.

Pour éviter les avertissements, vous devez désactiver les options suivantes :

- CPU Frequency scaling : CPU Power Management → CPU Frequency scaling → CPU Frequency scaling
- Allow for memory compaction : Memory Management options → Allow for memory compaction
- Contiguous Memory Allocator : Memory Management options → Contiguous Memory Allocator

On préfère aussi désactiver les options de débogage susceptibles d'augmenter la latence du système :

- Profiling support : General setup → Profiling support
- Tracers : Kernel hacking → Tracers
- KGDB-Kernel hacking : Kernel hacking → KGDB-Kernel hacking

Vous devrez aussi activer les PIPE pour l'échange entre tâches Linux/Xenomai, ainsi que les drivers temps-réel GPIO et/ou SPI :

- RTIPC protocol family : Xenomai/Cobalt → Drivers → Real-time IPC drivers → RTIPC protocol family

— User-space device driver framework : Xenomai/Cobalt → Drivers → UDD Support
— Real-time GPIO drivers : Xenomai/Cobalt → Drivers → Real-time GPIO drivers
— Real-time SPI masters drivers : Xenomai/Cobalt → Drivers → Real-time SPI masters drivers

3.2 Compilation du noyau

Cette étape peut durer un certain moment selon la puissance de votre machine hôte et le nombre de processeurs disponibles.
Pour lancer la compilation, il faut exécuter cette commande en choisissant le bon paramètre "*-jX*" en fonction du nombre de processeurs X dont dispose votre ordinateur hôte :

```
~/rpi-kernel/linux$ make -jX zImage
  SYSHDR arch/arm/include/generated/uapi/asm/unistd-common.h
SYSHDR arch/arm/include/generated/uapi/asm/unistd-oabi.h
WRAP arch/arm/include/generated/uapi/asm/bitsperlong.h
WRAP arch/arm/include/generated/uapi/asm/bpf_perf_event.h
WRAP arch/arm/include/generated/uapi/asm/errno.h
...
LDS arch/arm/boot/compressed/vmlinux.lds
AS arch/arm/boot/compressed/head.o
GZIP arch/arm/boot/compressed/piggy_data
CC arch/arm/boot/compressed/misc.o
CC arch/arm/boot/compressed/decompress.o
CC arch/arm/boot/compressed/string.o
AS arch/arm/boot/compressed/hyp-stub.o
AS arch/arm/boot/compressed/lib1funcs.o
AS arch/arm/boot/compressed/ashldi3.o
AS arch/arm/boot/compressed/bswapsdi2.o
AS arch/arm/boot/compressed/piggy.o
LD arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
```

Pour la suite de la compilation, il faut exécuter :

```
~/rpi-kernel/linux$ make -jX modules
CALL scripts/checksyscalls.sh
```

```
AS [M] arch/arm/crypto/aes-cipher-core.o
CC [M] arch/arm/crypto/aes-cipher-glue.o
AS [M] arch/arm/crypto/aes-neonbs-core.o
CC [M] arch/arm/crypto/aes-neonbs-glue.o
AS [M] arch/arm/crypto/sha1-armv4-large.o
CC [M] arch/arm/crypto/sha1_glue.o
AS [M] arch/arm/crypto/sha1-armv7-neon.o
CC [M] arch/arm/crypto/sha1_neon_glue.o
LD [M] arch/arm/crypto/aes-arm.o
LD [M] arch/arm/crypto/aes-arm-bs.o
LD [M] arch/arm/crypto/sha1-arm.o
GZIP kernel/config_data.gz
CC [M] mm/zsmalloc.o
...
LD [M] sound/soc/generic/snd-soc-audio-graph-card.ko
LD [M] sound/soc/generic/snd-soc-simple-card-utils.ko
LD [M] sound/soc/generic/snd-soc-simple-card.ko
LD [M] sound/soc/snd-soc-core.ko
LD [M] sound/usb/6fire/snd-usb-6fire.ko
LD [M] sound/usb/caiaq/snd-usb-caiaq.ko
LD [M] sound/usb/hiface/snd-usb-hiface.ko
LD [M] sound/usb/misc/snd-ua101.ko
LD [M] sound/usb/snd-usb-audio.ko
LD [M] sound/usb/snd-usbmidi-lib.ko


~/rpi-kernel/linux$ make -jX dtbs
  CALL scripts/checksyscalls.sh
DTC arch/arm/boot/dts/bcm2708-rpi-b.dtb
DTC arch/arm/boot/dts/bcm2708-rpi-b-plus.dtb
DTC arch/arm/boot/dts/bcm2708-rpi-cm.dtb
DTCO arch/arm/boot/dts/overlays/act-led.dtbo
DTC arch/arm/boot/dts/bcm2708-rpi-zero.dtb
DTC arch/arm/boot/dts/bcm2708-rpi-zero-w.dtb
DTC arch/arm/boot/dts/bcm2709-rpi-2-b.dtb
...
DTCO arch/arm/boot/dts/overlays/vc4-kms-kippah-7inch.dtbo
DTCO arch/arm/boot/dts/overlays/vc4-kms-v3d.dtbo
DTCO arch/arm/boot/dts/overlays/vga666.dtbo
```

```
42 DTCO arch/arm/boot/dts/overlays/w1-gpio.dtbo
43 DTCO arch/arm/boot/dts/overlays/w1-gpio-pullup.dtbo
44 DTCO arch/arm/boot/dts/overlays/w5500.dtbo
45 DTCO arch/arm/boot/dts/overlays/wittypi.dtbo
```

Nous allons maintenant préparer l'étape d'installation en exécutant :

```
1 ~/rpi-kernel/linux$ make -jX modules_install
2 INSTALL arch/arm/crypto/aes-arm-bs.ko
3 INSTALL arch/arm/crypto/aes-arm.ko
4 INSTALL arch/arm/crypto/sha1-arm-neon.ko
5 INSTALL arch/arm/crypto/sha1-arm.ko
6 INSTALL arch/arm/lib/xor-neon.ko
7 INSTALL crypto/af_alg.ko
8 INSTALL crypto/algif_skcipher.ko
9 INSTALL crypto/arc4.ko
10 INSTALL crypto/async_tx/async_memcpy.ko
11 ...
12 INSTALL sound/soc/generic/snd-soc-audio-graph-card.ko
13 INSTALL sound/soc/generic/snd-soc-simple-card-utils.ko
14 INSTALL sound/soc/generic/snd-soc-simple-card.ko
15 INSTALL sound/soc/snd-soc-core.ko
16 INSTALL sound/usb/6fire/snd-usb-6fire.ko
17 INSTALL sound/usb/caiaq/snd-usb-caiaq.ko
18 INSTALL sound/usb/hiface/snd-usb-hiface.ko
19 INSTALL sound/usb/misc/snd-ua101.ko
20 INSTALL sound/usb/snd-usb-audio.ko
21 INSTALL sound/usb/snd-usbmidi-lib.ko
22 DEPMOD 4.19.127-v7l+
23 Warning: modules_install: missing 'System.map' file. Skipping depmod.
24
25 ~/rpi-kernel/linux$ make -jX dtbs_install
26   INSTALL arch/arm/boot/dts/bcm2708-rpi-b-plus.dtb
27 INSTALL arch/arm/boot/dts/bcm2708-rpi-b.dtb
28 INSTALL arch/arm/boot/dts/bcm2708-rpi-cm.dtb
29 INSTALL arch/arm/boot/dts/bcm2708-rpi-zero-w.dtb
30 INSTALL arch/arm/boot/dts/bcm2708-rpi-zero.dtb
31 INSTALL arch/arm/boot/dts/bcm2709-rpi-2-b.dtb
32 INSTALL arch/arm/boot/dts/bcm2710-rpi-2-b.dtb
33 INSTALL arch/arm/boot/dts/bcm2710-rpi-3-b-plus.dtb
34 INSTALL arch/arm/boot/dts/bcm2710-rpi-3-b.dtb
```

```
INSTALL arch/arm/boot/dts/bcm2710-rpi-cm3.dtb
INSTALL arch/arm/boot/dts/bcm2711-rpi-4-b.dtb
...
INSTALL arch/arm/boot/dts/overlays/upstream.dtbo
INSTALL arch/arm/boot/dts/overlays/udrc.dtbo
INSTALL arch/arm/boot/dts/overlays/vc4-fkms-v3d.dtbo
INSTALL arch/arm/boot/dts/overlays/vc4-kms-kippah-7inch.dtbo
INSTALL arch/arm/boot/dts/overlays/vc4-kms-v3d.dtbo
INSTALL arch/arm/boot/dts/overlays/vga666.dtbo
INSTALL arch/arm/boot/dts/overlays/w1-gpio-pullup.dtbo
INSTALL arch/arm/boot/dts/overlays/w1-gpio.dtbo
INSTALL arch/arm/boot/dts/overlays/w5500.dtbo
INSTALL arch/arm/boot/dts/overlays/wittypi.dtbo
```

Une fois la compilation finie, vous devez finaliser l'image du noyau obtenue :

```
~/rpi-kernel/linux$ mkdir $INSTALL_MOD_PATH/boot

~/rpi-kernel/linux$ ./scripts/mkknlimg ./arch/arm/boot/zImage
                    $INSTALL_MOD_PATH/boot/$KERNEL.img
Version: Linux version 4.19.127-v7l+ (marwane@ubuntu)
         (gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2))
         #1 SMP Tue Sep 17 13:26:03 PDT 2024
DT: y
DDT: y
270x: y
283x: y
```

3.3 Transfert du noyau vers le Raspberry

Une fois la compilation terminée, vous devez compresser tous les fichiers pour les transférer sur le Raspberry Pi :

```
~/rpi-kernel/linux$ cd $INSTALL_MOD_PATH

~/rpi-kernel/rt-kernel$ tar czf ../xenomai-kernel.tgz *
```

Puis, transférez le noyau compressé vers le Raspberry en utilisant la commande *scp* :

```
~/rpi-kernel/rt-kernel$ cd ..

~/rpi-kernel$ scp xenomai-kernel.tgz pi@<ipaddress>:/tmp
```

Il faudra penser à remplacer *<ipaddress>* par l'IP de votre Raspberry Pi.

3.4 Mise en place du noyau sur le Raspberry

Maintenant, passez sur le Raspberry Pi :

```
1 ~/rpi-kernel$ ssh pi@<ipaddress>
```

Attention, il faudra penser à faire une sauvegarde de votre carte SD ou de vos données sur le Raspberry. Les commandes suivantes vont écraser l'ancien système ainsi qu'une partie des données.
Vous voilà prévenus, maintenant vous devez passer au déploiement du nouveau système. Pour cela tapez ces commandes :

```
1  ~$ cd /tmp
2
3  /tmp$ tar xzf xenomai-kernel.tgz
4
5  /tmp$ sudo cp *.dtb /boot/
6
7  /tmp$ cd boot
8
9  /tmp/boot$ sudo cp /boot/kernel7l.img /boot/kernel7l-backup.img
10
11 /tmp/boot$ sudo cp -rd * /boot/
12
13 /tmp/boot$ cd ../lib
14
15 /tmp/lib$ sudo cp -dr * /lib/
16
17 /tmp/lib$ cd ../overlays
18
19 /tmp/overlays$ sudo cp -d * /boot/overlays
20
21 /tmp/overlays$ cd ..
22
23 /tmp$ sudo cp -d bcm* /boot/
```

Il faut redémarrer le Raspberry Pi et si toutes les étoiles sont alignées, vous obtiendrez un noyau Xenomai fonctionnel :

```
1 ~$ sudo reboot
```

Pour preuve, exécutez cette commande pour vérifier :

```
~$ uname -a
Linux raspberrypi 4.19.127-v7l+ #1 SMP Tue Sep 17 13:26:03 PDT 2024
    armv7l GNU/Linux
```

Vous êtes bien passés sur le nouveau système "*Linux raspberrypi 4.19.127-v7+*".
Pour vérifier la version de Xenomai installée, exécutez :

```
~$ cat /proc/xenomai/version
3.1
```

Pour connaitre la version du *iPipe*, utilisez cette commande :

```
~$ cat /proc/ipipe/version
6
```

3.5 Compilation des outils Xenomai

Revenez sur votre PC maintenant afin de compiler les outils utilisables par Xenomai. En effet, Xenomai comprend une suite d'outils pour tester le fonctionnement du noyau temps réel, vous pouvez compiler ces outils en tapant :

```
~/rpi-kernel$ cd xenomai-3.1

~/rpi-kernel/xenomai-3.1$ ./scripts/bootstrap --with-core=cobalt
                    --enable-debug=partial
libtoolize: putting auxiliary files in AC_CONFIG_AUX_DIR, 'config'.
libtoolize: copying file 'config/ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIRS, 'config'.
libtoolize: copying file 'config/libtool.m4'
libtoolize: copying file 'config/ltoptions.m4'
libtoolize: copying file 'config/ltsugar.m4'
libtoolize: copying file 'config/ltversion.m4'
libtoolize: copying file 'config/lt~obsolete.m4'
configure.ac:80: installing 'config/compile'
configure.ac:105: installing 'config/missing'
demo/alchemy/Makefile.am: installing 'config/depcomp'


~/rpi-kernel/xenomai-3.1$ ./configure CFLAGS="-march=armv7-a
        -mfpu=vfp3" LDFLAGS="-mtune=cortex-a53"
        --build=i686-pc-linux-gnu --host=arm-linux-gnueabihf
```

```
21          --with-core=cobalt --enable-smp --enable-dlopen-libs
22          CC=${CROSS_COMPILE}gcc LD=${CROSS_COMPILE}ld
23 checking whether we build for Cobalt or Mercury core... cobalt
24 checking build system type... i686-pc-linux-gnu
25 checking host system type... arm-unknown-linux-gnueabihf
26 checking for a BSD-compatible install... /usr/bin/install -c
27 checking for arm-linux-gnueabihf-gcc... arm-linux-gnueabihf-gcc
28 checking whether the C compiler works... yes
29 checking for C compiler default output file name... a.out
30 checking for suffix of executables...
31 checking whether we are cross compiling... yes
32 checking for suffix of object files... o
33 ...
34 config.status: creating doc/Makefile
35 config.status: creating doc/doxygen/Makefile
36 config.status: creating doc/doxygen/xeno3prm-common.conf
37 config.status: creating doc/doxygen/xeno3prm-html.conf
38 config.status: creating doc/doxygen/xeno3prm-latex.conf
39 config.status: creating doc/gitdoc/Makefile
40 config.status: creating doc/asciidoc/Makefile
41 config.status: creating include/xeno_config.h
42 config.status: executing depfiles commands
43 config.status: executing libtool commands
```

Pour finir, on lance la compilation des librairies et des éxécutables :

```
1 ~/rpi-kernel/xenomai-3.1$ make -jX install DESTDIR=${PWD}/target
2 Making install in doc
3 make[1] : on entre dans le repertoire " /home/marwane/rpi-kernel/
4                           xenomai-3.1/doc "
5 Making install in gitdoc
6 make[2] : on entre dans le repertoire " /home/marwane/rpi-kernel/
7                           xenomai-3.1/doc/gitdoc "
8 make[3] : on entre dans le repertoire " /home/marwane/rpi-kernel/
9                           xenomai-3.1/doc/gitdoc "
10 make[3]: rien a faire pour " install-exec-am ".
11 make[3]: rien a faire pour " install-data-am ".
12 ...
13 make[1] : on entre dans le repertoire " /home/marwane/rpi-kernel/
14                           xenomai-3.1 "
15 make[2] : on entre dans le repertoire " /home/marwane/rpi-kernel/
```

```
16                      xenomai-3.1 "
17 if test -r /home/marwane/rpi-kernel/xenomai-3.1/target//usr/xenomai/
18                      etc/udev/udev.rules ; then \
19 for f in ./kernel/cobalt/udev/*.rules ; do \
20 b=`basename $f` ; \
21 grep -q Xenomai:`basename $b .rules` /home/marwane/rpi-kernel/
22           xenomai-3.1/target//usr/xenomai/etc/udev/udev.rules || \
23 ( echo ; cat $f ) >> /home/marwane/rpi-kernel/xenomai-3.1/target/
24                                  usr/xenomai/etc/udev/udev.rules ; \
25 done ; \
26 else \
27 /bin/bash /home/marwane/rpi-kernel/xenomai-3.1/config/install-sh -d
28 /home/marwane/rpi-kernel/xenomai-3.1/target//usr/xenomai/etc/udev/rules.d; \
29 for f in ./kernel/cobalt/udev/*.rules ; do \
30 /usr/bin/install -c -m 644 $f /home/marwane/rpi-kernel/xenomai-3.1/
31                                  target//usr/xenomai/etc/udev/rules.d/ ; \
32 done ; \
33 fi
34 make[2]: rien a faire pour " install-data-am ".
35 make[2] : on quitte le repertoire " /home/marwane/rpi-kernel/xenomai-3.1 "
36 make[1] : on quitte le repertoire " /home/marwane/rpi-kernel/xenomai-3.1 "
```

3.6 Transfert des outils Xenomai vers le Raspberry

Une fois la compilation terminée, vous devez compresser tous les fichiers compilés des outils Xenomai pour les transférer sur le Raspberry Pi :

```
1 ~/rpi-kernel/xenomai-3.1$ cd target
2
3 ~/rpi-kernel/xenomai-3.1/target$ tar czf ../../xenomai-tools.tgz *
```

Puis, transférez le noyau compressé vers le Raspberry en utilisant la commande *scp* :

```
1 ~/rpi-kernel/xenomai-3.1$ cd ~/rpi-kernel/
2
3 ~/rpi-kernel$ scp xenomai-tools.tgz pi@<ipaddress>:/tmp
```

De même, il faudra penser à remplacer ***<ipaddress>*** par l'IP correspondante de votre Raspberry Pi.

3.7 Mise en place des outils Xenomai sur le Raspberry

Maintenant, repassez sur le Raspberry Pi et exécutez ces commandes afin de déployer les outils de Xenomai :

```
~$ cd /tmp

/tmp$ sudo tar xzf xenomai-tools.tgz

/tmp$ sudo cp -r usr/* /usr/
```

4 Test du système temps-réel Xenomai

Maintenant que Xenomai est bien installé, vous êtes capables de tester son bon fonctionnement grâce aux outils que vous venez de rajouter.
Pour cela, vous pouvez exécuter les suites de tests comme par exemple :

```
~$ cd /usr/xenomai/bin/

/usr/xenomai/bin/$ ls

/usr/xenomai/bin/$ sudo ./latency
== Sampling period: 1000 us
== Test mode: periodic user-mode task
== All results in microseconds
warming up...
RTT|  00:00:01 (periodic user-mode task, 1000 us period, priority 99)
RTH|-----lat min|-----lat avg|-----lat max|-overrun|
---msw|---lat best|--lat worst
RTD|  -4.315|  -4.119|   0.037|       0|       0|  -4.315|   0.037
RTD|  -4.297|  -4.097|   0.555|       0|       0|  -4.315|   0.555
RTD|  -4.168|  -3.758|   0.110|       0|       0|  -4.315|   0.555
RTD|  -4.280|  -4.085|  -0.132|       0|       0|  -4.315|   0.555
RTD|  -4.188|  -2.075|  26.775|       0|       0|  -4.315|  26.775
RTD|  -4.281|  -3.403|  14.441|       0|       0|  -4.315|  26.775
RTD|  -3.893|  -0.883|  25.570|       0|       0|  -4.315|  26.775
...
```