



UNIVERSITÉ DE TECHNOLOGIE DE
COMPIÈGNE

MI11

Rapport des TPs linux embarqué

Clément BLANQUET et Rafik CHENNOUF

Juin 2017

Sommaire

1	Rapport TP 1 - Linux embarqué	5
	Exercice 1 : Prise en main de la carte DevKit8600	5
	1.1 Question 1.1	5
	1.2 Question 1.2	5
	1.3 Question 1.3	5
	1.4 Question 1.4	5
	1.5 Question 1.5	6
	Exercice 2	6
	2.1 Question 2.1	6
	2.2 Question 2.2	6
	2.3 Question 2.3	7
	Exercice 3	8
	3.1 Question 3.1	8
	3.2 Question 3.2	8
	3.3 Question 3.3	8
	3.4 Question 3.4	8
	3.5 Question 3.5	10
	Exercice 4 : Ajout de paquets	10
	4.1 Question 4.1	10
	4.2 Question 4.2	12
	4.3 Question 4.3	12
	4.4 Question 4.4	13
	Exercice 5 : Compilation manuelle du noyau	13
	5.1 Question 5.1	13
	5.2 Question 5.2	14
	5.3 Question 5.3	14
	5.4 Question 5.4	14
	5.5 Question 5.5	17
	5.6 Question 5.6	17
	5.7 Question 5.7	19
2	Rapport TP 2 - Linux embarqué	20
	Exercice 1 : Hello World	20
	1.1 Question 1.1	20

1.2	Question 1.2	20
1.3	Question 1.3	20
1.4	Question 1.4	21
Exercice 2	: Clignotement des LEDs	21
2.1	Question 2.1	21
2.2	Question 2.2	22
2.3	Question 2.3	22
Exercice 3	: Boutons poussoirs	24
3.1	Question 3.1	24
3.2	Question 3.2	24
3.3	Question 3.3	24
3.4	Question 3.4	24
Exercice 4	: Charge CPU	27
4.1	Question 4.1	27
4.2	Question 4.2	27
3	TP1 Xenomai	30
Exercice 1	: tâches	30
1.1	Question 1.1	30
1.2	Question 1.2	30
1.3	Question 1.3	31
1.4	Question 1.4	32
1.5	Question 1.5	32
Exercice 2	: Synchronisation	33
2.1	Question 2.1	33
2.2	Question 2.2	35
2.3	Question 2.3	35
2.4	Question 2.4	35
2.5	Question 2.5	35
2.6	Question 2.6	37
2.7	Question 2.7	40
Exercice 3	: Latence	41
3.1	Question 3.1	41
3.2	Question 3.2	42
3.3	Question 3.3	44
4	TP2 Xenomai	45
Exercice	: Pathfinder	45
1.1	Question 1	45
1.2	Question 2	45
1.3	Question 3	46

1.4	Question 4	46
1.5	Question 5	47
1.6	Question 6	48
A	Messages de sortie du terminal entre l'allumage de la cible et le prompt de login	49

Table des figures

1.1	Dossier image	7
1.2	Mise en évidence du port série dans /proc/devices	9
1.3	Mise en évidence du port série dans /dev	10
1.4	Dossier <i>ipk</i>	10
1.5	Dossier <i>ipk/devkit800</i>	11
1.6	Dossier <i>ipk/armv7a-vfp-neon</i>	12
1.7	Dossier <i>/usr/lib</i>	13
1.8	Configurations par défaut de <i>devkit8600</i>	14
1.9	Activation du driver pour les LEDs	15
1.10	Logs de démarrage	17
1.11	Logs du kernel	19
2.1	Commande <i>file</i>	20
2.2	Commande <i>source</i>	20
2.3	Commande <i>file</i>	21
2.4	Résultat <i>Hello World</i>	21
2.5	Fichiers des LEDs	22
2.6	Manipulation des LEDs	22
2.7	Résultat du evtest sur /dev/input/event1	24
3.1	Statistiques Xenomai avec un programme non temps réel	30
3.2	Statistiques Xenomai avec une tâche temps réel	32
3.3	Statistiques Xenomai avec une tâche temps réel et la fonction <code>rt_task_sleep</code>	32
3.4	Statistiques Xenomai avec une tâche temps réel et les fonctions <code>rt_task_sleep</code> et <code>rt_printf</code>	33
3.5	Fichier de statistiques de Xenomai	40
3.6	Fichier du scheduler de Xenomai	40
3.7	Résultat du programme <i>Latence</i> sans stress	44
3.8	Résultat du programme <i>Latence</i> avec stress	44
4.1	Résultat avec <code>ORDO_BUS</code>	47
4.2	Enchaînement avec meilleur cas pour METEO (40ms)	48
4.3	Enchaînement avec pire cas pour METEO (60ms)	48

Rapport TP 1 - Linux embarqué

Exercice 1 : Prise en main de la carte DevKit8600

1.1 Question 1.1

La DevKit8600 est basée sur les processeurs AM3359 de Texas Instrument. Elle possède un ARM Cortex-A8 cadencé à 720 MHz avec un boot ROM On-Chip de 176KB. Elle possède un certain nombre d'interfaces comme un port LAN, une entrée/sortie audio, des USB, une interface JTAG... etc.

1.2 Question 1.2

A FAIRE

1.3 Question 1.3

Nous avons utilisé une liaison série pour nous connecter avec l'application cutecom.

Voici les paramètres que l'on a utilisés :

1. Bits per second : 115200
2. Data bits : 8
3. Parity : None
4. Stop bits : 1
5. Flow Control : None

1.4 Question 1.4

On constate qu'il y a une erreur au démarrage.

```
1 TFTP error: 'File not found' (1)
3 ERROR: can't get kernel image!
```

La cible ne peut donc pas démarrer. Le fichier manquant est l'image du kernel, appelé **uImage**. Il faudrait la placer dans le dossier `/tftpboot` sur notre PC, puisque la cible nous indique au démarrage qu'elle va chercher l'image à cet endroit.

1.5 Question 1.5

Même si l'image était présente, la cible ne pourrait pas démarrer car il manque un rootfs (système de fichier racine).

Exercice 2

2.1 Question 2.1

Le dossier `/home/mi11/poky/build/conf` contient les fichiers de configurations de poky qui permettent donc de configurer l'image selon nos besoins :

- `bblayers.conf`
- `local.conf`
- `sanity_info`
- `templateconf.cfg`

Le dossier `/home/mi11/devkit8600/meta-devkit8600` contient les fichiers spécifiques à notre cible qui vont permettre de construire une image qui lui est adaptée.

2.2 Question 2.2

Nous avons ajouté une ligne dans le fichier `bblayers.conf` :

```
1 BBLAYERS ?= " \
  /home/mi11/poky-dizzy -12.0.3/meta \
3 /home/mi11/poky-dizzy -12.0.3/meta-yocto \
  /home/mi11/poky-dizzy -12.0.3/meta-yocto-bsp \
5 /home/mi11/devkit8600/meta-devkit8600 \ // CELLE LA
  "
```

De plus, à la ligne 36 du fichier `local.conf`, nous avons inscrit :

```
MACHINE ??= "devkit8600"
```

Le nom `"devkit8600"` est en fait le nom du fichier de configuration du même nom situé ici : `/home/mi11/devkit8600/meta-devkit8600/conf/machine`, ce qui fait donc le lien entre la génération de l'image et les paramètres de la cible.

```

mi11@mi11-VirtualBox ~/poky/build/tmp/deploy/images $ ll devkit8600/
total 79736
drwxr-xr-x 2 mi11 mi11 4096 avril 14 15:13 ./
drwxr-xr-x 3 mi11 mi11 4096 avril 14 15:07 ../
-rw-r--r-- 1 mi11 mi11 5442 avril 14 15:12 core-image-base-devkit8600-20170414130644.rootfs.manifest
-rw-r--r-- 1 mi11 mi11 18255233 avril 14 15:13 core-image-base-devkit8600-20170414130644.rootfs.tar.bz2
-rw-r--r-- 1 mi11 mi11 30670848 avril 14 15:13 core-image-base-devkit8600-20170414130644.rootfs.ubi
-rw-r--r-- 1 mi11 mi11 29458432 avril 14 15:13 core-image-base-devkit8600-20170414130644.rootfs.ubifs
lrwxrwxrwx 1 mi11 mi11 57 avril 14 15:13 core-image-base-devkit8600.manifest -> core-image-base-devkit8600-20170414130644.rootfs.manifest
lrwxrwxrwx 1 mi11 mi11 56 avril 14 15:13 core-image-base-devkit8600.tar.bz2 -> core-image-base-devkit8600-20170414130644.rootfs.tar.bz2
lrwxrwxrwx 1 mi11 mi11 52 avril 14 15:13 core-image-base-devkit8600.ubi -> core-image-base-devkit8600-20170414130644.rootfs.ubi
lrwxrwxrwx 1 mi11 mi11 54 avril 14 15:13 core-image-base-devkit8600.ubifs -> core-image-base-devkit8600-20170414130644.rootfs.ubifs
-rw-r--r-- 1 mi11 mi11 21617 avril 14 15:07 modules--3.1.0-r0-devkit8600-20170410160636.tgz
lrwxrwxrwx 1 mi11 mi11 47 avril 14 15:07 modules-devkit8600.tgz -> modules--3.1.0-r0-devkit8600-20170410160636.tgz
-rw-r--r-- 1 mi11 mi11 294 avril 14 15:11 README - DO NOT DELETE FILES IN THIS DIRECTORY.txt
-rw-r--r-- 1 mi11 mi11 192 avril 14 15:13 ubinize.cfg
lrwxrwxrwx 1 mi11 mi11 46 avril 14 15:07 uImage -> uImage--3.1.0-r0-devkit8600-20170410160636.bin
-rw-r--r-- 1 mi11 mi11 3215152 avril 14 15:07 uImage--3.1.0-r0-devkit8600-20170410160636.bin
lrwxrwxrwx 1 mi11 mi11 46 avril 14 15:07 uImage-devkit8600.bin -> uImage--3.1.0-r0-devkit8600-20170410160636.bin
mi11@mi11-VirtualBox ~/poky/build/tmp/deploy/images $

```

FIGURE 1.1 – Dossier image

2.3 Question 2.3

Voici ce qu'on obtient dans le dossier `/home/mi11/poky/build/tmp/deploy/images` après la compilation :

On obtient notre image kernel *uImage* qui est en fait un lien symbolique vers le fichier *uImage-3.1.0-r0-devkit8600-20170410160636.bin*.

On voit également le système de fichier nommé *rootfs* qui est compressé dans une archive bzip2.

Si on compare la taille des fichiers qui viennent d'être générés avec ceux de notre VM sous Linux Mint :

- CIBLE :
 - Taille image : 3,2 MB
 - Taille rootfs : 1,8 MB
- HOTE :
 - Taille image : 6,6 MB (on le voit dans `/boot`)
 - Taille système de fichiers : environ 7 GB

On constate une bonne différence entre les deux images car l'image de la cible n'est faite que pour cette cible là, elle gère moins de choses et propose moins de fonctionnalités que celle de notre VM. Pour le système de fichiers, on constate une énorme différence car beaucoup plus de programmes sont installés sur notre VM par rapport à la cible sur laquelle rien ou presque n'est installé. La cible dispose en fait de la configuration minimale.

Exercice 3

3.1 Question 3.1

Comme nous l'avions indiqué dans le premier exercice, nous avons copié uImage dans le dossier `/tftpboot`, comme c'est indiqué dans le fichier de configuration `/etc/xinetd.d/tftp`. Le rootfs, lui, est copié dans le dossier `/tftpboot/rootfs`

3.2 Question 3.2

Messages de sortie du terminal entre l'allumage de la cible et le prompt de login : voir annexe A.1.

A FINIR

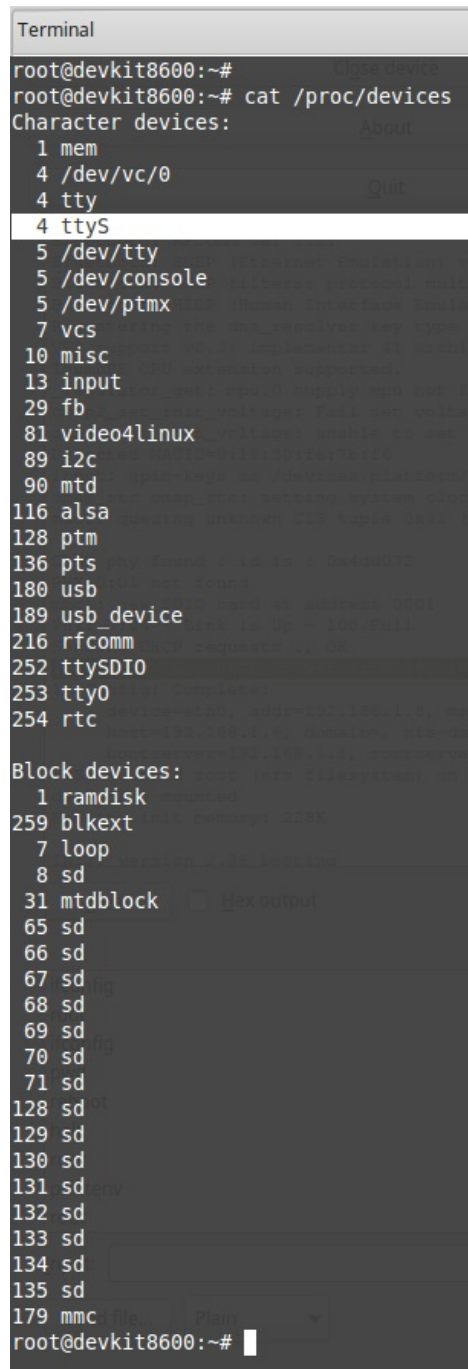
3.3 Question 3.3

L'IP de la cible est 192.168.1.6. On peut voir cette adresse sur les messages de sortie du boot : *"IP-Config : Got DHCP answer from 192.168.1.1, my address is 192.168.1.6 (sortie du boot)"*

3.4 Question 3.4

`proc/devices` contient les périphériques (dans la première section) ainsi que les stockages (dans la deuxième section).

Le numéro en début de ligne dans le fichier `/proc/devices` correspond au numéro mineur, qui indique si les périphériques sont gérés par le même driver.



```
Terminal
root@devkit8600:~#
root@devkit8600:~# cat /proc/devices
Character devices:
1 mem
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
7 vcs
10 misc
13 input
29 fb
81 video4linux
89 i2c
90 mtd
116 alsa
128 ptm
136 pts
180 usb
189 usb_device
216 rfcomm
252 ttyS0
253 tty0
254 rtc
Block devices:
1 ramdisk
259 blkext
7 loop
8 sd
31 mtdblock
65 sd
66 sd
67 sd
68 sd
69 sd
70 sd
71 sd
128 sd
129 sd
130 sd
131 sd
132 sd
133 sd
134 sd
135 sd
179 mmcblk0
root@devkit8600:~#
```

FIGURE 1.2 – Mise en évidence du port série dans `/proc/devices`

```
root@devkit8600:~# ls /dev/ttyS
ttyS0 ttyS1 ttyS2 ttyS3
root@devkit8600:~# ls /dev/ttyS
```

FIGURE 1.3 – Mise en évidence du port série dans /dev

3.5 Question 3.5

A FAIRE

Exercice 4 : Ajout de paquets

4.1 Question 4.1

```
mi11@mi11-VirtualBox ~/poky/build/tmp/deploy/ipk $ ll
total 480
drwxr-xr-x 6 mi11 mi11  4096 avril 14 15:30 ./
drwxr-xr-x 6 mi11 mi11  4096 avril 14 15:29 ../
drwxr-xr-x 2 mi11 mi11  4096 avril 14 15:29 all/
drwxr-xr-x 2 mi11 mi11 294912 avril 14 15:30 armv7a-vfp-neon/
drwxr-xr-x 2 mi11 mi11  4096 avril 14 15:29 devkit8600/
-rw-r--r-- 1 mi11 mi11    0 avril 14 15:11 Packages
drwxr-xr-x 2 mi11 mi11 172032 avril 14 15:29 x86_64-native_sdk/
```

FIGURE 1.4 – Dossier *ipk*

Le dossier `/home/mi11/poky/build/tmp/deploy/ipk` est organisé par architectures (x86/x64, devkit8600, armv7a, etc.).

```

mill@mill-VirtualBox ~/poky/build/tmp/deploy/ipk/devkit8600 $ ll
total 36764
drwxr-xr-x 2 mill mill 4096 avril 14 15:29 ./
drwxr-xr-x 6 mill mill 4096 avril 14 15:30 ../
-rw-r--r-- 1 mill mill 4046 avril 14 15:08 base-files_3.0.14-r89_devkit8600.ipk
-rw-r--r-- 1 mill mill 840 avril 14 15:08 base-files-dbg_3.0.14-r89_devkit8600.ipk
-rw-r--r-- 1 mill mill 872 avril 14 15:08 base-files-dev_3.0.14-r89_devkit8600.ipk
-rw-r--r-- 1 mill mill 922 avril 14 15:08 base-files-doc_3.0.14-r89_devkit8600.ipk
-rw-r--r-- 1 mill mill 1072 avril 14 15:07 depmodwrapper-cross_1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 704 avril 14 15:07 depmodwrapper-cross-dbg_1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 736 avril 14 15:07 depmodwrapper-cross-dev_1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 2722 avril 14 15:07 kernel-3.1.0_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 678 avril 14 15:07 kernel_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 30265112 avril 14 15:07 kernel-dev_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 3195142 avril 14 15:07 kernel-image-3.1.0_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 702 avril 14 15:07 kernel-modules_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 1922 avril 14 15:07 kernel-module-scsi-wait-scan_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 19494 avril 14 15:07 kernel-module-usbserial_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 3939110 avril 14 15:07 kernel-vmlinux_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 826 avril 14 15:08 opkg-config-base_1.0-r1_devkit8600.ipk
-rw-r--r-- 1 mill mill 682 avril 14 15:08 opkg-config-base-dbg_1.0-r1_devkit8600.ipk
-rw-r--r-- 1 mill mill 714 avril 14 15:08 opkg-config-base-dev_1.0-r1_devkit8600.ipk
-rw-r--r-- 1 mill mill 776 avril 14 15:07 packagegroup-base_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 692 avril 14 15:07 packagegroup-base-3g_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 726 avril 14 15:07 packagegroup-base-bluetooth_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 790 avril 14 15:07 packagegroup-base-dbg_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 822 avril 14 15:07 packagegroup-base-dev_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 660 avril 14 15:07 packagegroup-base-extended_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 668 avril 14 15:07 packagegroup-base-ipv6_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 688 avril 14 15:07 packagegroup-base-nfc_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 690 avril 14 15:07 packagegroup-base-nfs_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 712 avril 14 15:07 packagegroup-base-usb-gadget_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 742 avril 14 15:07 packagegroup-base-usb-host_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 712 avril 14 15:07 packagegroup-base-vfat_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 772 avril 14 15:07 packagegroup-base-wifi_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 680 avril 14 15:07 packagegroup-base-zeroconf_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 734 avril 14 15:07 packagegroup-core-boot_1.0-r17_devkit8600.ipk
-rw-r--r-- 1 mill mill 804 avril 14 15:07 packagegroup-core-boot-dbg_1.0-r17_devkit8600.ipk
-rw-r--r-- 1 mill mill 832 avril 14 15:07 packagegroup-core-boot-dev_1.0-r17_devkit8600.ipk
-rw-r--r-- 1 mill mill 684 avril 14 15:07 packagegroup-distro-base_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 722 avril 14 15:07 packagegroup-machine-base_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 26783 avril 14 15:29 Packages
-rw-r--r-- 1 mill mill 4259 avril 14 15:29 Packages.gz
-rw-r--r-- 1 mill mill 2558 avril 14 15:29 Packages.stamps
-rw-r--r-- 1 mill mill 868 avril 14 15:07 poky-feed-config-opkg_1.0-r2_devkit8600.ipk
-rw-r--r-- 1 mill mill 684 avril 14 15:07 poky-feed-config-opkg-dbg_1.0-r2_devkit8600.ipk
-rw-r--r-- 1 mill mill 714 avril 14 15:07 poky-feed-config-opkg-dev_1.0-r2_devkit8600.ipk
-rw-r--r-- 1 mill mill 1450 avril 14 15:09 shadow-securetty_4.2.1-r3_devkit8600.ipk
-rw-r--r-- 1 mill mill 690 avril 14 15:09 shadow-securetty-dbg_4.2.1-r3_devkit8600.ipk


```

FIGURE 1.5 – Dossier *ipk/devkit800*

Les sous-dossiers contiennent des paquets d'extension *.ipk*. Le noyau se trouve dans le sous-dossier *devkit8600* comme nous pouvons le voir sur la copie d'écran ci-dessus. Quant au paquet *libxml2*, il se trouve dans le sous-dossier *armv7a-vfp-neon*.

4.2 Question 4.2

Les différents paquets relatifs à *libxml2* sont visibles sur la copie d'écran suivante.



```
mi11@mi11-VirtualBox ~/poky/build/tmp/deploy/ipk/armv7a-vfp-neon $ ls | grep libxml2
libxml2_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-dbg_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-dev_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-doc_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-ptest_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-staticdev_2.9.1-r0_armv7a-vfp-neon.ipk
mi11@mi11-VirtualBox ~/poky/build/tmp/deploy/ipk/armv7a-vfp-neon $
```

FIGURE 1.6 – Dossier *ipk/armv7a-vfp-neon*

Nous avons ensuite copié le fichier

```
libxml2_2.9.1-r0_armv7a-vfp-neon.ipk
```

dans le système de fichiers de la cible et nous avons installé le paquet avec la commande suivante :

```
opkg install libxml2_2.9.1-r0_armv7a-vfp-neon.ipk
```

4.3 Question 4.3

La première méthode pour copier le fichier dans le système de fichiers de la cible consiste à utiliser directement la commande *cp* comme suit :

```
sudo cp libxml2_2.9.1-r0_armv7a-vfp-neon.ipk /tftpboot/rootfs/home/root/
```

La deuxième méthode consiste à copier le fichier à distance via la commande *scp* qui se base sur *ssh* comme suit :

```
scp libxml2_2.9.1-r0_armv7a-vfp-neon.ipk root@192.168.1.6:/home/root/
```

4.4 Question 4.4

Les fichiers de *libxml2* installés par le gestionnaire de paquets se trouvent dans le dossier */usr/lib* de notre cible.

```
root@devkit8600:~# ls /usr/lib/
dbus/          libgio-2.0.so.0.4000.0  libnl-genl-3.so.200
gio/           libglib-2.0.so.0        libnl-genl-3.so.200.20.0
libX11.so.6    libglib-2.0.so.0.4000.0  libnl-nf-3.so.200
libX11.so.6.3.0 libgmodule-2.0.so.0      libnl-nf-3.so.200.20.0
libXau.so.6     libgmodule-2.0.so.0.4000.0  libnl-route-3.so.200
libXau.so.6.0.0 libgmp.so.10            libnl-route-3.so.200.20.0
libXdmcpc.so.6 libgmp.so.10.2.0        libopkg.so.1
libXdmcpc.so.6.0.0 libgnutls.so.28         libopkg.so.1.0.0
libavahi-common.so.3 libgnutls.so.28.38.0    libpyglib-2.0-python.so.0
libavahi-common.so.3.5.3 libgobject-2.0.so.0     libpyglib-2.0-python.so.0.0.0
libavahi-core.so.7 libgobject-2.0.so.0.4000.0  libpython2.7.so.1.0
libavahi-core.so.7.0.2 libgpg-error.so.0       libreadline.so.6
libbluetooth.so.3 libgpg-error.so.0.10.0  libreadline.so.6.3
libbluetooth.so.3.13.0 libgthread-2.0.so.0     libssl.so.1.0.0
libdaemon.so.0 libgthread-2.0.so.0.4000.0  libtirpc.so.1
libdaemon.so.0.5.0 libhistory.so.6         libtirpc.so.1.0.10
libdbus-1.so.3  libhistory.so.6.3       libxcb.so.1
libdbus-1.so.3.8.4 libhogweed.so.2        libxcb.so.1.1.0
libdbus-glib-1.so.2 libhogweed.so.2.5      libxml2.so.2
libdbus-glib-1.so.2.2.2 libkmod.so.2           libxml2.so.2.9.1
libexpat.so.1  libkmod.so.2.2.8       locale/
libexpat.so.1.6.0 libnettle.so.4         neard/
libffi.so.6    libnettle.so.4.7       opkg/
libffi.so.6.0.2 libnl-3.so.200         python2.7/
libgcrypt.so.20 libnl-3.so.200.20.0    ssl/
libgcrypt.so.20.0.1 libnl-cli-3.so.200
libgio-2.0.so.0 libnl-cli-3.so.200.20.0
```

FIGURE 1.7 – Dossier */usr/lib*

Ce dossier contient des fichiers *shared object* (.so) qui sont des librairies/bibliothèques partagées dynamiques. Les .so sont des bibliothèques qui se chargent en mémoire au moment de l'exécution d'un programme qui les utilisent. Si plusieurs programmes les utilisant sont lancés en même temps, une seule instance de la bibliothèque dynamique réside en mémoire.

Exercice 5 : Compilation manuelle du noyau

5.1 Question 5.1

La page 72 de la documentation nous dit qu'il est possible d'allumer et d'éteindre les LEDs via les commandes suivantes :

```
root@DevKit8600:~# echo 1 > /sys/class/leds/user_led/brightness
root@DevKit8600:~# echo 0 > /sys/class/leds/user_led/brightness
```

Comme le dossier

user_led

n'existe pas ce n'est pas opérationnel.

5.2 Question 5.2

Nous avons ensuite compiler manuellement le noyau via la chaîne de compilation croisée en exécutant le script suivant :

```
source /opt/poky/1.7.3/environment-setup-armv7a-vfp-neon-poky-linux-gnueabi
unset LDFLAGS
```

Le fichier ci-dessus sert à mettre en place l'environnement de compilation de manière à avoir les bons préfixes pour une compilation croisée. Le préfixe la chaîne de compilation croisée est *arm-poky-linux-gnueabi*.

5.3 Question 5.3

Pour obtenir la liste des configurations par défaut du noyau pour une architecture ARM, il faut utiliser la commande *make ARCH=arm help*. Les configurations par défaut possibles pour la carte *devkit8600* sont visibles sur la copie d'écran ci-dessous :

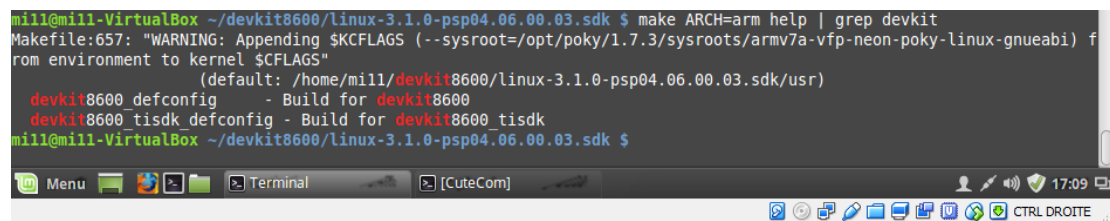


FIGURE 1.8 – Configurations par défaut de *devkit8600*

Nous avons retenu uniquement la première configuration et nous l'avons mise en place via la commande suivante :

```
make devkit8600_defconfig
```

5.4 Question 5.4

Ensuite, nous avons lancé la personnalisation du noyau via la commande suivante : *make ARCH=arm menuconfig*. Le but étant d'ajouter un driver pour les LEDs connectées par GPIO. Pour cela, nous avons activé l'option *LED Support for GPIO connected LEDs* comme on peut le voir sur la copie d'écran suivante.

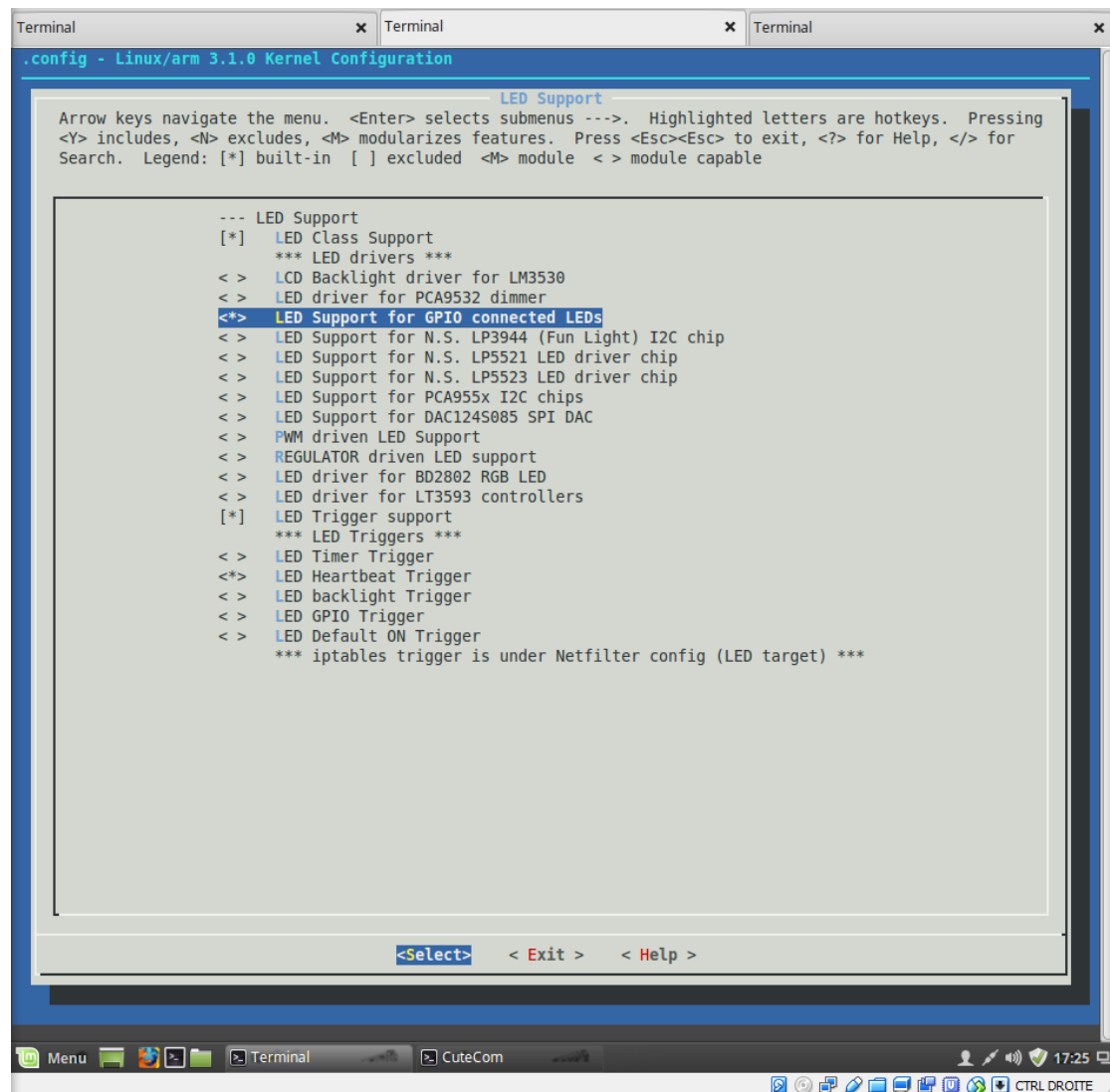


FIGURE 1.9 – Activation du driver pour les LEDs

Ce driver peut être activé via deux modes, *modularizes features* ou *built-in*. Le premier mode est utilisé si l'on souhaite ajouter un driver en tant que module qui sera chargé en mémoire uniquement en cas de besoin et déchargé lorsque le kernel n'en a pas plus besoin. Ceci est utile lorsque l'on souhaite avoir un kernel pas très lourd. Quant au deuxième mode, le driver sera directement intégré au kernel et il sera disponible tout le temps. Le kernel sera donc plus gros, plus lent et utilisera plus de mémoire. Dans notre cas, nous avons utilisé le mode *built-in*.

Nous avons ensuite compiler notre noyau avec la commande suivante : *make*

ARCH=arm uImage -j2.

5.5 Question 5.5

Le résultat de la compilation se trouve dans *arch/arm/boot/uImage*.

Nous avons ensuite copier le fichier uImage dans /tftpboot pour qu'il soit utilisé par la cible puis nous l'avons démarré.

5.6 Question 5.6

Pour vérifier dans les logs de démarrage que le noyau utilisé est bien celui qui vient d'être compilé, il suffit de lire la date de compilation :

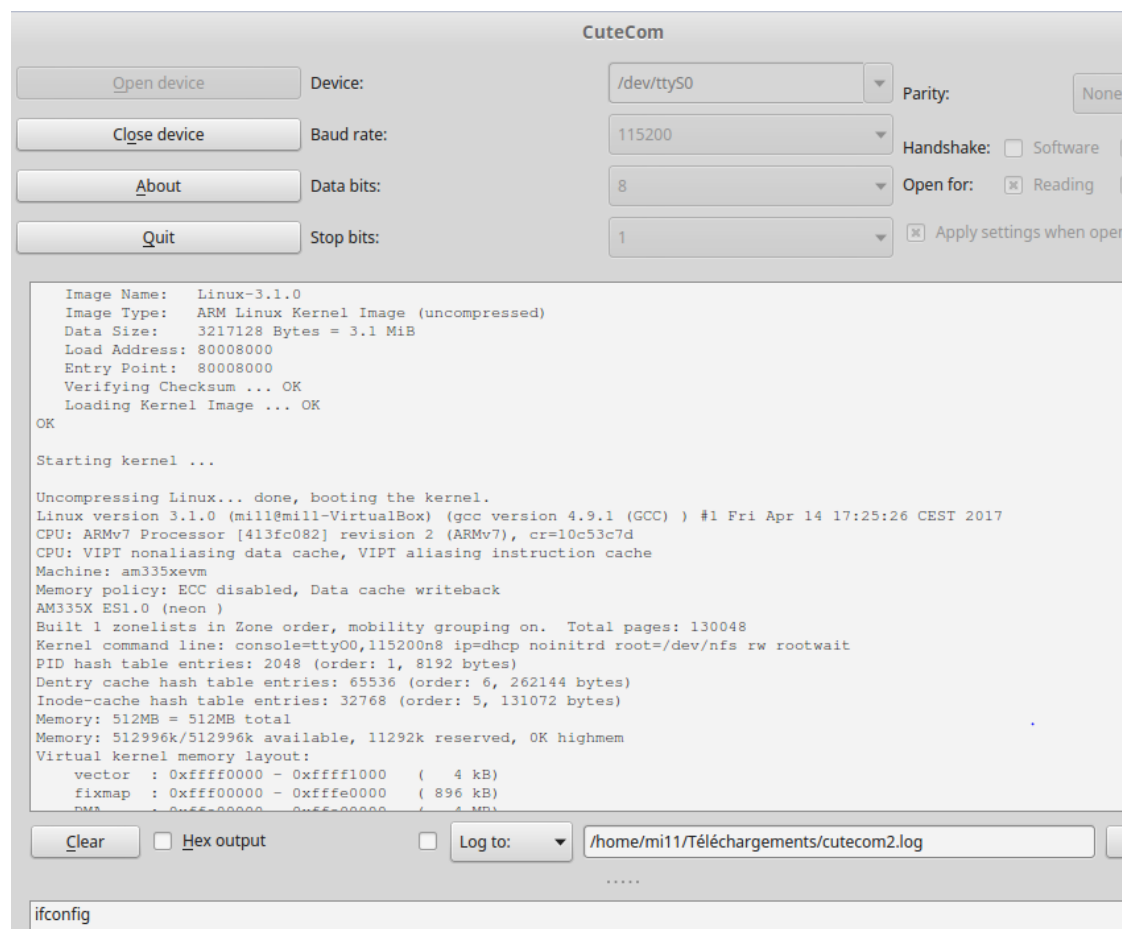


FIGURE 1.10 – Logs de démarrage

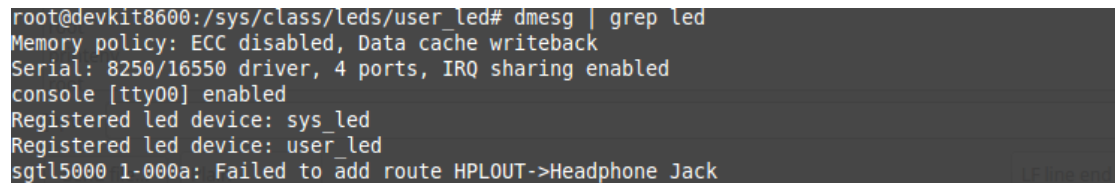
Nous pouvons lire sur la copie d'écran précédente la ligne suivante :

Linux version 3.1.0 (mi11@mi11-VirtualBox) (gcc version 4.9.1 (GCC)) #1 Fri Apr 14

La date de compilation du noyau correspond bien à celui qui vient d'être compilé.

5.7 Question 5.7

Pour vérifier dans les logs de démarrage que la fonctionnalité ajoutée est bien présente, on peut utiliser la commande *dmesg* pour afficher les messages du kernel :



```
root@devkit8600:/sys/class/leds/user_led# dmesg | grep led
Memory policy: ECC disabled, Data cache writeback
Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
console [tty00] enabled
Registered led device: sys_led
Registered led device: user_led
sgtl5000 1-000a: Failed to add route HPL0UT->Headphone Jack
```

FIGURE 1.11 – Logs du kernel

Le driver a bien été ajouté et le dossier

`used_led`

a été créé.

Rapport TP 2 - Linux embarqué

Exercice 1 : Hello World

1.1 Question 1.1

Après avoir compilé notre programme *Hello World* avec **gcc** nous avons exécuté la commande *file* pour obtenir des informations sur l'exécutable. Nous nous sommes rendu compte que cet exécutable a été compilé pour une architecture classique x86/64 et non ARM. C'est pour cela que le programme ne peut pas s'exécuter sur la cible.

```
mill@mill-VirtualBox ~/Documents/tp6 $ file main
main: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=41339e3d9c8d801a732fb7004c7bc66e29f5eaca, not stripped
```

FIGURE 2.1 – Commande *file*

1.2 Question 1.2

Avant de pouvoir cross-compiler notre programme il faut activer l'environnement de cross-compilation via la commande *source* sur le fichier `/opt/poky/1.7.3/environment-setup-armv7a-vfp-neon-poky-linux-gnueabi`.

```
mill@mill-VirtualBox ~ $ source /opt/poky/1.7.3/environment-setup-armv7a-vfp-neon-poky-linux-gnueabi
mill@mill-VirtualBox ~ $ echo $CC
arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=softfp -mfpu=neon --sysroot=/opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi
```

FIGURE 2.2 – Commande *source*

1.3 Question 1.3

En utilisant de nouveau la commande *file* après la cross-compilation, nous constatons que le nouveau exécutable a bien été compilé pour une architecture ARM.

```

mill@mill-VirtualBox ~/Documents/tp6 $ gcc -o main main.c
mill@mill-VirtualBox ~/Documents/tp6 $ ls
main  main.c
mill@mill-VirtualBox ~/Documents/tp6 $ file main
main: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked (u
ses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=87a3c670badb9a2b8a6e1debc66fb
67b52fbbc8e, not stripped

```

FIGURE 2.3 – Commande *file*

1.4 Question 1.4

Voici le programme *Hello World* :

```

1 #include <stdio.h>
3 int main()
4 {
5     printf("Hello World !");
6     return 0;
7 }

```

Ainsi que le résultat :

```

mill@mill-VirtualBox ~/Documents/tp6 $ ssh root@192.168.1.6
root@devkit8600:~# ls
libxml2_2.9.1-r0_armv7a-vfp-neon.ipk  main
root@devkit8600:~# ./main
Hello World !root@devkit8600:~#

```

FIGURE 2.4 – Résultat *Hello World*

Exercice 2 : Clignotement des LEDs

2.1 Question 2.1

Pour accéder aux LEDs il suffit de manipuler les fichiers
 /sys/class/leds/user_led/brightness
 pour la LED utilisateur et
 /sys/class/leds/sys_led/brightness
 pour la LED système.

```
root@devkit8600:~# cat /sys/class/leds/user_led/brightness
0
root@devkit8600:~# cat /sys/class/leds/sys_led/brightness
0
```

FIGURE 2.5 – Fichiers des LEDs

2.2 Question 2.2

L'allumage d'une LED se fait en écrivant le caractère '1' dans le fichier correspondant et l'éteignage se fait en écrivant le caractère '0' comme on peut le voir sur la copie d'écran suivante.

```
root@devkit8600:~# echo 1 > /sys/class/leds/sys_led/brightness
root@devkit8600:~# echo 0 > /sys/class/leds/sys_led/brightness
root@devkit8600:~# echo 1 > /sys/class/leds/user_led/brightness
root@devkit8600:~# echo 0 > /sys/class/leds/user_led/brightness
```

FIGURE 2.6 – Manipulation des LEDs

2.3 Question 2.3

Le programme suivant allume en alternance la LED utilisateur et la LED système chaque seconde :

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <fcntl.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6
7 int main()
8 {
9     int sys_led, user_led;
10
11     sys_led = open("/sys/class/leds/sys_led/brightness", O_RDWR);
12     // ouverture des fichiers
13     user_led = open("/sys/class/leds/user_led/brightness", O_RDWR);
14
15     if(sys_led == -1 || user_led == -1)
16     {
17         perror("Erreur d'ouverture de l'un des fichiers");
18         return 0;
19     }
```

```
21 while(1)
22 {
23     if(write(user_led, "0", 1) == -1) // éteingage de la LED user
24         perror("Erreur d'écriture");
25
26     if(write(sys_led, "1", 1) == -1) // allumage de la LED système
27         perror("Erreur d'écriture");
28
29     sleep(1); // attente d'une seconde
30
31     if(write(sys_led, "0", 1) == -1) // éteingage de la LED système
32         perror("Erreur d'écriture");
33
34     if(write(user_led, "1", 1) == -1) // allumage de la LED user
35         perror("Erreur d'écriture");
36
37     sleep(1); // attente d'une seconde
38 }
39
40 close(sys_led); // fermeture
41 close(user_led); // des fichiers
42
43 return 0;
44 }
```



```

root@devkit8600:~# evtest /dev/input/event1
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 1 (KEY_ESC)
    Event code 59 (KEY_F1)
    Event code 102 (KEY_HOME)
Properties:
Testing ... (interrupt to exit)
Event: time 1492179011.627030, type 1 (EV_KEY), code 1 (KEY_ESC), value 1
Event: time 1492179011.627034, ----- SYN REPORT -----
Event: time 1492179011.806361, type 1 (EV_KEY), code 1 (KEY_ESC), value 0
Event: time 1492179011.806365, ----- SYN REPORT -----
Event: time 1492179020.838352, type 1 (EV_KEY), code 102 (KEY_HOME), value 1
Event: time 1492179020.838356, ----- SYN REPORT -----
Event: time 1492179021.034120, type 1 (EV_KEY), code 102 (KEY_HOME), value 0
Event: time 1492179021.034122, ----- SYN REPORT -----
Event: time 1492179022.975083, type 1 (EV_KEY), code 59 (KEY_F1), value 1
Event: time 1492179022.975091, ----- SYN REPORT -----
Event: time 1492179023.172103, type 1 (EV_KEY), code 59 (KEY_F1), value 0
Event: time 1492179023.172106, ----- SYN REPORT -----

```

FIGURE 2.7 – Résultat du evtest sur /dev/input/event1

Exercice 3 : Boutons poussoirs

3.1 Question 3.1

La cible dispose de trois boutons utilisateur (HOME, BACK, MENU) et un bouton de reset. On y accède via le fichier `"/dev/input/event1"` dans lequel toutes les informations quant aux pressions et relâchement de ces boutons seront écrites.

3.2 Question 3.2

Pour tester en ligne de commande, on peut utiliser `"evtest /dev/input/event1"` (voir figure 2.7)

Les valeurs sont soit à 1 soit à 0 selon l'état du bouton : pressé ou relâché.

3.3 Question 3.3

La structure `input_event` est déclarée dans le fichier suivant :
`" /opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi/usr/include/linux/input.h"`

Au début de notre fichier C, on ajoute donc : `"#include <linux/input.h>".` Voici le programme que nous avons codé. Celui-ci lit les événements indiqués dans le fichier `"/dev/input/event1"`, affiche ces événements dans le terminal et allume :

- La `sys_led` en cas de pression sur le bouton MENU
- La `user_led` en cas de pression sur le bouton BACK
- Les deux LEDs en cas de pression sur la touche HOME

3.4 Question 3.4

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <fcntl.h>
4 #include <linux/input.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7
8 int main()
9 {
10     struct input_event event_buttons;
11     int sys_led, user_led, buttons;
12
13     buttons = open("/dev/input/event1", O_RDONLY);
14
15     if(buttons == -1)
16     {
17         perror("Erreur d'ouverture de l'un des fichiers");
18         return 0;
19     }
20
21     sys_led = open("/sys/class/leds/sys_led/brightness", O_RDWR);
22     user_led = open("/sys/class/leds/user_led/brightness", O_RDWR);
23
24     if(sys_led == -1 || user_led == -1)
25     {
26         perror("Erreur d'ouverture de l'un des fichiers");
27         return 0;
28     }
29
30     while(1)
31     {
32         read(buttons, &event_buttons, sizeof(event_buttons));
33
34         int code = (int)event_buttons.code;
35         int value = (int)event_buttons.value;
36
37         switch(code)
38         {
39             case KEY_F1: // MENU
40                 if(value == 1)
41                 {
42                     printf("MENU PUSH\n");
43                     if(write(sys_led, "1", 1) == -1)
44                         perror("Erreur d'écriture");
45                 }
46                 else
47                 {
48
```

```
50         printf("MENU RELEASE\n");
51         if(write(sys_led, "0", 1) == -1)
52             perror("Erreur d'écriture");
53     }
54     break;
55
56     case KEY_ESC: // BACK
57         if(value == 1)
58         {
59             printf("BACK PUSH\n");
60             if(write(user_led, "1", 1) == -1)
61                 perror("Erreur d'écriture");
62         }
63         else
64         {
65             printf("BACK RELEASE\n");
66             if(write(user_led, "0", 1) == -1)
67                 perror("Erreur d'écriture");
68         }
69         break;
70
71     case KEY_HOME: // HOME
72         if(value == 1)
73         {
74             printf("HOME PUSH\n");
75             if(write(user_led, "1", 1) == -1)
76                 perror("Erreur d'écriture");
77             if(write(sys_led, "1", 1) == -1)
78                 perror("Erreur d'écriture");
79         }
80         else
81         {
82             printf("HOME RELEASE\n");
83             if(write(user_led, "0", 1) == -1)
84                 perror("Erreur d'écriture");
85             if(write(sys_led, "0", 1) == -1)
86                 perror("Erreur d'écriture");
87         }
88         break;
89     }
90 }
91 if(close(sys_led) == -1)
92     perror("Erreur de fermeture de l'un des fichiers");
93 if(close(user_led) == -1)
94     perror("Erreur de fermeture de l'un des fichiers");
95 if(close(buttons) == -1)
96     perror("Erreur de fermeture de l'un des fichiers");
97
98 return 0;
```

98 }

Exercice 4 : Charge CPU

4.1 Question 4.1

Voici notre programme :

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <fcntl.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6 #include <sys/time.h>
7
8 #define TAILLE 10000
9
10 int main()
11 {
12     int j = 0, tempstotal = 0;
13     struct timeval debut, fin, res;
14     gettimeofday(&debut, NULL);
15     for(j = 0; j < TAILLE; j++)
16     {
17         usleep(1000);
18     }
19     gettimeofday(&fin, NULL);
20     timersub(&fin, &debut, &res);
21     int tempstotal = res.tv_sec* 1000000 + res.tv_usec;
22
23     printf("Temps total : %d\n", tempstotal);
24
25     return 0;
26 }
```

A COMPLETER

4.2 Question 4.2

Voici notre programme modifié :

```
1 #include <stdio.h>
2 #include <unistd.h>
```

```
#include <fcntl.h>
4 #include <sys/types.h>
#include <sys/stat.h>
6 #include <sys/time.h>

8 #define TAILLE 10000

10 int main()
{
12     int j = 0, min, max, moy = 0, tempstotal = 0;
    int tab[TAILLE];
14     struct timeval debut, fin, res;

16     for(j = 0; j < TAILLE; j++)
    {
18         gettimeofday(&debut, NULL);
        usleep(1000);
20         gettimeofday(&fin, NULL);
        timersub(&fin, &debut, &res);
22         tab[j] = res.tv_sec * 1000000 + res.tv_usec;
        moy += tab[j];
24     }

26     moy = moy / TAILLE - 1000;
    min = tab[0];
28     max = tab[0];

30     for(j = 1; j < TAILLE; j++)
    {
32         if(min > tab[j])
            min = tab[j];

34         if(max < tab[j])
            max = tab[j];
36     }

38     min -= 1000;
40     max -= 1000;

42     printf("Moyenne : %d\nMaximum : %d\nMinimum : %d\n", moy, max, min)
        ;

44     return 0;
}
```

Voici les temps relevés :

— Sans stress :

- Moyenne : 85 ms
- Maximum : 772 ms
- Minimum : 13 ms
- Avec stress :
 - Moyenne : 766 ms
 - Maximum : 304397 ms
 - Minimum : 27 ms

TP1 Xenomai

Exercice 1 : tâches

1.1 Question 1.1

Un code "classique" ne s'exécute pas de façon temps réel. En effet, il n'apparaît pas dans le fichier `/proc/xenomai/stats` ce qui signifie qu'il n'est pas temps réel.

```
root@devkit8600-xenomai:~# cat /proc/xenomai/stat
CPU  PID  MSW   CSW   PF   STAT   %CPU  NAME
0  0      0      0      0  00500080  100.0  ROOT
0  0      0  9592      0  00000000   0.0  IRQ68: [timer]
```

FIGURE 3.1 – Statistiques Xenomai avec un programme non temps réel

1.2 Question 1.2

Voici le code qui permet de créer une tâche temps réel :

```
1 #include <stdio.h>
2 #include <native/task.h>
3 #include <analogy/os_facilities.h>
4 #include <unistd.h>
5 #include <sys/mman.h>
6
7 #define TASK_PRIO 99
8 #define TASK_MODE 0
9 #define TASK_STKSZ 0
10
11 RT_TASK task_printf;
12
13 void task_hello()
14 {
15     while(1)
16     {
17         sleep(rt_timer_ns2ticks(1000000000));
18         printf("Hello World !\n");
19     }
20 }
21
```

```
int main()  
23 {  
    mlockall(MCL_CURRENT|MCL_FUTURE);  
25    rt_print_auto_init(1);  
    int err;  
27  
    err = rt_task_create(&task_printf, "hello world", TASK_STKSZ,  
        TASK_PRIO, TASK_MODE);  
29    if (!err)  
        rt_task_start(&task_printf, &task_hello, NULL);  
31  
    getchar();  
33  
    return 0;  
35 }
```

Le chemin des fichiers à inclure est :

```
1 /opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi/usr/  
    include/xenomai
```

Le chemin des bibliothèques est :

```
1 /opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi/usr/lib
```

Cela nous donne la ligne de commande (après avoir fait un *source*) :

```
1 $CC -o main main.c -I/opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-  
    linux-gnueabi/usr/include/xenomai -L/opt/poky/1.7.3/sysroots/  
    armv7a-vfp-neon-poky-linux-gnueabi/usr/lib -lxenomai -lnative
```

1.3 Question 1.3

Cette application n'est toujours pas temps réel. En effet, malgré la création d'une tâche temps réel, les fonctions *sleep* et *printf* qui s'y trouvent ne sont pas temps réel.

Le fichier de statistiques Xenomai donne :


```

root@devkit8600-xenomai:~# cat /proc/xenomai/stat
CPU  PID  MSW   CSW   PF   STAT  %CPU  NAME
 0    0    0     335   0    00500080 100.0  ROOT
 0    0    0    11084  0    00000000  0.0  IRQ68: [timer]

```

FIGURE 3.2 – Statistiques Xenomai avec une tâche temps réel

1.4 Question 1.4

Voici le code après avoir remplacé la fonction sleep par son équivalent temps réel :

```

1 void task_hello()
2 {
3     while(1)
4     {
5         rt_task_sleep(rt_timer_ns2ticks(1000000000));
6         printf("Hello World !\n");
7     }
8 }

```

Le fichier de statistiques Xenomai donne :

```

root@devkit8600-xenomai:~# cat /proc/xenomai/stat
CPU  PID  MSW   CSW   PF   STAT  %CPU  NAME
 0    0    0     202   0    00500080 100.0  ROOT
 0   965    4      9   0    00300184  0.0  hello world
 0    0    0    7802  0    00000000  0.0  IRQ68: [timer]

```

FIGURE 3.3 – Statistiques Xenomai avec une tâche temps réel et la fonction rt_task_sleep

1.5 Question 1.5

Voici le code après avoir remplacé les fonctions sleep et printf par leurs équivalents temps réel :

```

1 void task_hello()
2 {
3     while(1)
4     {
5         rt_task_sleep(rt_timer_ns2ticks(1000000000));
6         rt_printf("Hello World !\n");
7     }
8 }

```

Le fichier de statistiques Xenomai donne :

```
root@devkit8600-xenomai:~# cat /proc/xenomai/stat
CPU  PID    MSW      CSW      PF    STAT      %CPU  NAME
0     0       0        46       0     00500080  100.0  ROOT
0     957     0        46       0     00300184  0.0    hello world
0     0       0       2860     0     00000000  0.0    IRQ68: [timer]
```

FIGURE 3.4 – Statistiques Xenomai avec une tâche temps réel et les fonctions `rt_task_sleep` et `rt_printf`

Interprétation des statistiques Xenomai : On constate que :

- Lorsqu'on crée une tâche temps réel qui n'utilise aucune fonction temps réel (dans notre exemple, les fonctions *sleep* et *printf*, elle n'apparaît pas dans le fichier de statistiques. Elle n'est en fait absolument pas temps réel.
- Lorsque, dans cette même tâche, on rend la fonction *sleep* temps réel (en la remplaçant par *rt_task_sleep*), elle apparaît dans le fichier de statistiques. On constate que son nombre de changements de contexte (9) est très inférieur au nombre de changements de contexte du ROOT (282), ce qui signifie que ... A COMPLETER
- Lorsque les fonctions *sleep* et *printf* ont été remplacées par leurs équivalents temps réel (*rt_task_sleep* et *rt_printf*), le nombre changements de contextes de la tâche est égal à celui de ROOT (46), ce qui montre alors que la tâche est "entièrement" temps réel.

Exercice 2 : Synchronisation

2.1 Question 2.1

Voici le code du programme lançant deux tâches Xenomai qui afficheront chacune une partie du message "Hello World!" :

```
#include <stdio.h>
2 #include <native/task.h>
#include <analogy/os_facilities.h>
4 #include <unistd.h>
#include <sys/mman.h>
6
#define TASK_PRIO 99
8 #define TASK_MODE 0
#define TASK_STKSZ 0
```

```
10 RT_TASK task_printf;
12 RT_TASK task_printf2;

14 void task_hello()
15 {
16     rt_printf("Hello\n");
17 }

18 void task_world()
19 {
20     rt_printf("World !\n");
22     rt_task_sleep(rt_timer_ns2ticks(1000000000));
23 }

24

26 int main()
27 {
28     mlockall(MCL_CURRENT|MCL_FUTURE);
29     rt_print_auto_init(1);
30     int err1, err2;

32     err1 = rt_task_create(&task_printf, "hello", TASK_STKSZ, TASK_PRIO,
33                          TASK_MODE);
34     err2 = rt_task_create(&task_printf2, "world", TASK_STKSZ, TASK_PRIO,
35                          TASK_MODE);
36     if (!err1 && !err2)
37     {
38         rt_task_start(&task_printf, &task_hello, NULL);
39         rt_task_join(&task_printf);
40         rt_task_start(&task_printf2, &task_world, NULL);
41         rt_task_join(&task_printf2);
42     }

42     getchar();

44     return 0;
45 }
```

Le résultat est le suivant :

```
1 root@devkit8600-xenomai:~# ./synchro
Hello
3 World !
c
5 root@devkit8600-xenomai:~#
```

2.2 Question 2.2

Pour le moment, les priorités des tâches n'ont aucune influence. En effet, elles sont lancées dans l'ordre dans lequel elles se trouvent dans notre code : après le `rt_task_start`, rien ne "bloque" l'exécution de la tâche ; les tâches se lancent donc l'une après l'autre comme on l'a défini. Pour afficher les messages de le désordre, il faut donc inverser les deux tâches le code. On peut aussi utiliser des sémaphores, comme vu dans les questions suivantes.

2.3 Question 2.3

Il faut initialiser le sémaphore à 0 de manière à bloquer les tâches.

2.4 Question 2.4

Le paramètre *mode* lors de la création du sémaphore nous sert à définir le mode d'ordonnancement à utiliser. Par exemple, si on choisit le mode *S_FIFO* alors les tâches seront ordonnancés en suivant la méthode FIFO (First In First Out), c'est à dire que les tâches attendront dans leur ordre d'arrivée que le sémaphore se libère. Un second exemple de *mode* utilisable est le mode *S_PRIO* qui permet d'ordonner les tâches par ordre de priorité, c'est à dire que les tâches avec la plus haute priorité auront accès au sémaphore avant les tâches de plus faible priorité.

2.5 Question 2.5

```
1 #include <stdio.h>
  #include <native/task.h>
3 #include <analogy/os_facilities.h>
  #include <unistd.h>
5 #include <sys/mman.h>
  #include <native/sem.h>
7
  #define TASK_PRIO_HELLO 98
9 #define TASK_PRIO_WORLD 99
  // #define TASK_PRIO 99
11 #define TASK_MODE 0
  #define TASK_STKSZ 0
13
14 RT_TASK task_printf;
15 RT_TASK task_printf2;
  RT_SEM sem;
17
```

```
19 void task_hello()      // affichage de 'Hello'
20 {
21     rt_sem_p (&sem, 0); // décrémentation du sémaphore
22     rt_printf("Hello\n");
23 }
24
25 void task_world()      // affichage de 'World'
26 {
27     rt_sem_p (&sem, 0); // décrémentation du sémaphore
28     rt_printf("World !\n");
29 }
30
31 int main()
32 {
33     mlockall(MCL_CURRENT|MCL_FUTURE);
34     rt_print_auto_init(1);
35     int err1, err2;
36
37     // création du sémaphore en mode FIFO
38     rt_sem_create (&sem, "sem", 0, S_FIFO);
39
40     // création des deux tâches
41     err1 = rt_task_create(&task_printf, "hello", TASK_STKSZ,
42                          TASK_PRIO_HELLO, TASK_MODE);
43     err2 = rt_task_create(&task_printf2, "world", TASK_STKSZ,
44                          TASK_PRIO_WORLD, TASK_MODE);
45
46     if (!err1 && !err2)
47     {
48         // démarrage de la tâche qui affiche 'Hello'
49         rt_task_start(&task_printf, &task_hello, NULL);
50
51         // attente de sa terminaison
52         rt_task_join(&task_printf);
53
54         // démarrage de la tâche qui affiche 'World'
55         rt_task_start(&task_printf2, &task_world, NULL);
56
57         // attente de sa terminaison
58         rt_task_join(&task_printf2);
59
60         getchar();
61
62         rt_sem_v (&sem); // incrémentation du sémaphore
63         rt_sem_v (&sem); // incrémentation du sémaphore
64     }
65 }
```

```
67 }  
    return 0;
```

Le programme précédent avec le mode *S_FIFO* pour le sémaphore nous donne le résultat suivant :

```
1 root@devkit8600-xenomai:~# ./synchro  
3 Hello  
   World !
```

Cependant, si nous utilisons le mode *S_PRIO* pour le sémaphore nous obtenons :

```
root@devkit8600-xenomai:~# ./synchro  
2  
   World !  
4 Hello
```

Ceci est cohérent puisque nous avons défini les priorités comme suit :

```
1 #define TASK_PRIO_HELLO 98  
2 #define TASK_PRIO_WORLD 99
```

La tâche qui s'occupe d'afficher le 'World!' est plus prioritaire que la tâche qui affiche le 'Hello'.

2.6 Question 2.6

```
#include <stdio.h>  
2 #include <native/task.h>  
  #include <analogy/os_facilities.h>  
4 #include <unistd.h>  
  #include <sys/mman.h>  
6 #include <native/sem.h>  
  
8 #define TASK_PRIO_HELLO 98  
  #define TASK_PRIO_WORLD 97  
10 #define TASK_PRIO_METRO 99  
   // #define TASK_PRIO 99  
12 #define TASK_MODE 0  
   #define TASK_STKSZ 0
```

```
14 RT_TASK task_printf;
16 RT_TASK task_printf2;
17 RT_TASK task_metronome;
18 RT_SEM sem;

20 void task_hello()
21 {
22     while(1)
23     {
24         rt_sem_p (&sem, 0); // décrémentation du sémaphore
25         rt_printf("Hello\n");
26     }
27 }

28 void task_world()
29 {
30     while(1)
31     {
32         rt_sem_p (&sem, 0); // décrémentation du sémaphore
33         rt_printf("World !\n");
34         rt_sem_v (&sem); // incrémentation du sémaphore
35     }
36 }

38 void task_metro() // tâche métronome
39 {
40     while(1)
41     {
42         rt_sem_v (&sem); // incrémentation du sémaphore
43         rt_sem_v (&sem); // incrémentation du sémaphore

44         rt_sem_p (&sem, 0); // décrémentation du sémaphore
45         rt_task_sleep(rt_timer_ns2ticks(1000000000)); // attente 1 sec
46     }
47 }

50

52 int main()
53 {
54     mlockall(MCL_CURRENT|MCL_FUTURE);
55     rt_print_auto_init(1);
56     int err1, err2, err3;

58     // création du sémaphore en mode PRIO
59     rt_sem_create (&sem, "sem", 0, S_PRIO);

60     err1 = rt_task_create(&task_printf, "hello", TASK_STKSZ,
61                          TASK_PRIO_HELLO, TASK_MODE);
```

```
62 err2 = rt_task_create(&task_printf2, "world", TASK_STKSZ,  
TASK_PRIO_WORLD, TASK_MODE);  
  
64 // création de la tâche métronome  
err3= rt_task_create(&task_metronome, "metro", TASK_STKSZ,  
TASK_PRIO_METRO, TASK_MODE);  
  
66 if (!err1 && !err2 && !err3)  
68 {  
    rt_task_start(&task_printf, &task_hello, NULL);  
70    rt_task_join(&task_printf);  
    rt_task_start(&task_printf2, &task_world, NULL);  
72    rt_task_join(&task_printf2);  
  
74    // démarrage de la tâche métronome  
    rt_task_start(&task_metronome, &task_metro, NULL);  
76    // attente de sa terminaison  
    rt_task_join(&task_metronome);  
  
78    getchar();  
  
80 }  
  
82 return 0;  
84 }
```

Le programme ci-dessus nous donne le résultat suivant à l'infini :

```
root@devkit8600-xenomai:~# ./synchro  
2 Hello  
4 World !  
6 Hello  
8 World !  
Hello  
World !  
Hello  
World !
```

La tâche métronome possède la plus haute priorité comme on peut le voir ci-dessous :

```
#define TASK_PRIO_HELLO 98  
2 #define TASK_PRIO_WORLD 97  
#define TASK_PRIO_METRO 99
```


Initialement, les tâches 'Hello' et 'World' sont bloquées. A chaque activation de la tâche métronome, le sémaphore est incrémenté de 2. A la première incrémentation, c'est la tâche 'Hello' qui prend la main puisque elle a une priorité supérieure à la tâche 'World'. La tâche 'Hello' décrémente le sémaphore et affiche 'Hello'. Ensuite, la tâche métronome reprend son exécution puisque la tâche 'World' est bloquée et elle incrémente une seconde fois le sémaphore. La tâche 'World' est donc débloquée, elle décrémente le sémaphore, affiche 'World!' puis incrémente le sémaphore pour débloquer la tâche métronome qui va reprendre son exécution en décrémentant le sémaphore et en faisant une attente d'une seconde avant de relancer le même processus. *** CLEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEMENT : TU PEUX CONFIRMER CE QUE JE DIS J'AI UN DOUTE ***

2.7 Question 2.7

Le fichier de statistiques de Xenomai lors du lancement de notre programme est le suivant :

```
root@devkit8600-xenomai:~# cat /proc/xenomai/stat
*CPU  PID    MSW      CSW      PF      STAT      %CPU  NAME
0  0      0        513      0      00500080  99.8  ROOT
0  1264   0        43       0      00300182  0.0   hello
0  1265   0        85       0      00300182  0.0   world
0  1266   0        84       0      00300184  0.0   metro
0  0      0        53562    0      00000000  0.0   IRQ68: [timer]
```

FIGURE 3.5 – Fichier de statistiques de Xenomai

On retrouve bien nos 3 tâches 'hello', 'world' et 'metro'. On voit également que ces tâches ne subissent que très peu de changements de contexte (CSW) car ce sont des tâches temps-réel.

Pour le fichier du scheduler de Xenomai nous avons :

```
root@devkit8600-xenomai:~# cat /proc/xenomai/sched
CPU  PID    CLASS  PRI      TIMEOUT  TIMEBASE  STAT      NAME
0  0      idle   -1       -         master    R         ROOT
0  1264   rt     98       -         master    W         hello
0  1265   rt     97       -         master    W         world
0  1266   rt     99       625ms 11us master    D         metro
```

FIGURE 3.6 – Fichier du scheduler de Xenomai

Nous retrouvons bien les priorités de nos tâches que nous avons défini dans notre code. On voit également que nos trois tâches sont temps-réel via la colonne *CLASS* (rt). On voit également que la tâche 'ROOT' qui correspond à Linux est en mode 'idle', c'est à dire qu'elle a priorité la plus faible (-1). Dans la colonne *STAT*, on remarque que la tâche 'metro' est marquée de la lettre 'D' qui signifie 'Delayed', c'est à dire que la tâche est retardée sans aucune autre condition d'attente (wait d'une seconde). Les tâches 'hello' et 'metro' sont marquées de la lettre 'W' qui signifie que ces tâches sont en attente d'une ressource (sémaphore) et la tâche 'ROOT' est marquée de la lettre 'R' qui signifie 'Runnable', c'est à dire que la tâche est exécutable.

Exercice 3 : Latence

3.1 Question 3.1

```
1 #include <stdio.h>
2 #include <native/task.h>
3 #include <analogy/os_facilities.h>
4 #include <unistd.h>
5 #include <sys/mman.h>
6 #include <native/sem.h>
7 #include <nucleus/timer.h>
8
9 #define TASK_PRIO 99
10 #define TASK_MODE 0
11 #define TASK_STKSZ 0
12 #define TAILLE 10000
13
14 RT_TASK task_latence;
15
16 void task_wait()
17 {
18     int i;
19
20     // boucle de 1 à 10000
21     for(i = 0; i < TAILLE; i++)
22     {
23         // attente de 1ms
24         rt_task_sleep(rt_timer_ns2ticks(1000000));
25     }
26 }
27
28 int main()
29 {
30     mlockall(MCL_CURRENT|MCL_FUTURE);
```

```
31  rt_print_auto_init(1);
    int err1;
33
    err1 = rt_task_create(&task_latence, "wait", TASK_STKSZ, TASK_PRIO,
        TASK_MODE);
35
    if(!err1)
37    {
        rt_task_start(&task_latence, &task_wait, NULL);
39        rt_task_join(&task_latence);

41        getchar();
    }
43
45  return 0;
}
```

3.2 Question 3.2

```
#include <stdio.h>
2 #include <native/task.h>
#include <analogy/os_facilities.h>
4 #include <unistd.h>
#include <sys/mman.h>
6 #include <native/sem.h>
#include <nucleus/timer.h>
8

10 #define TASK_PRIO 99
#define TASK_MODE 0
12 #define TASK_STKSZ 0
#define TAILLE 10000
14
RT_TASK task_latence;
16

void task_wait()
18 {
    int i;
20    RTIME moy = 0, min, max;
    RTIME begin, end;
22
    min = 999999999999999;
24    max = 0;
26
```

```
28  for(i = 0; i < TAILLE; i++)
    {
30      begin = rt_timer_read(); // lecture du temps
      rt_task_sleep(rt_timer_ns2ticks(1000000));
      end = rt_timer_read(); // lecture du temps
32
      if(min > end - begin) // calcul du min
34          min = end - begin;

36      if(max < end - begin) // calcul du max
      {
38          max = end - begin;
      }

40      moy += end - begin; // calcul de la moyenne
42  }

44  moy = moy / TAILLE;

46  rt_printf("Moyenne :%llu \nMaximum : %llu \nMinimum : %llu\n", moy,
      max, min);

48  }

50
51  int main()
52  {
53      mlockall(MCL_CURRENT|MCL_FUTURE);
54      rt_print_auto_init(1);
55      int err1;

56      err1 = rt_task_create(&task_latence, "wait", TASK_STKSZ, TASK_PRIO,
          TASK_MODE);

58
59      if(!err1)
60      {
61          rt_task_start(&task_latence, &task_wait, NULL);
62          rt_task_join(&task_latence);

64          getchar();
65      }

66
67
68      return 0;
    }
```

Le programme précédent nous donne le résultat suivant :

```
root@devkit8600-xenomai:~# ./latence
Moyenne :1002756
Maximum : 1044920
Minimum : 1002440
```

FIGURE 3.7 – Résultat du programme *Latence* sans stress

3.3 Question 3.3

Ici, on charge le CPU via la commande *stress* avant de lancer notre programme :

```
^Croot@devkit8600-xenomai:~# ./latence
Moyenne :1005789
Maximum : 1030760
Minimum : 1003560
```

FIGURE 3.8 – Résultat du programme *Latence* avec stress

On remarque que les résultats avec charge CPU et sans charge CPU sont pratiquement les mêmes ce qui veut dire que la charge n'a aucune influence sur les tâches exécutées. Ceci est logique puisque le programme est temps-réel donc une charge CPU ne ralentira que les programmes non temps-réel qui sont toujours reliés à Linux (ROOT) comme on a pu le constater dans le tp précédent. Ici, les tâches temps-réel ont la plus grande priorité sur le CPU.

TP2 Xenomai

Exercice : Pathfinder

1.1 Question 1

La fonction *create_and_start_rt_task...*

La fonction *rt_task...*

La structure *task_descriptor* permet d'obtenir des informations de la tâche en cours, c'est à dire :

- la tâche elle même (de type `RT_TASK`)
- le pointeur vers la fonction associée
- la période de la tâche
- la durée d'exécution la tâche
- la priorité de la tâche
- si la tâche utilise une ressource ou non

1.2 Question 2

La fonction *rt_task_name* sert à obtenir le nom de la tâche en cours grâce à la structure `RT_TASK_INFO` et son champ *name*. Cette structure a d'autres champs :

- *bprio* : priorité de base (ne change pas au cours du temps)
- *cprio* : priorité actuelle (peut changer au cours du temps)
- *status* : statut de la tâche
- *relpoint* : temps restant avant la prochaine exécution
- *exectime* : temps d'exécution de la tâche depuis son lancement
- *modeswitches* : nombre de changements de mode primaire / secondaire
- *ctxswitches* : nombre de changements de contextes
- *pagefaults* : nombre de défauts de page

1.3 Question 3

Voici notre fonction *busy_wait* :

```
1 void busy_wait(RTIME time)
  {
3     static RT_TASK_INFO info;      // info sur la tâche
     rt_task_inquire(NULL,&info);    // initialisation des infos dont
     exectime
5     RTIME begin = info.exectime;    // recuperation du temps d'
     execution initial (> 0)

7
     while(info.exectime - begin < time)
9         rt_task_inquire(NULL,&info); // initialisation des infos
     dont exectime
  }
```

On commence par acquérir les informations relatives à la tâche grâce à une structure `RT_TASK_INFO` et à la fonction `rt_task_inquire`. La champ *exectime* de cette structure nous permet d'obtenir le temps d'exécution de la tâche depuis son lancement (donc il y a certainement eu plusieurs occurrences). On appelle ce temps *begin*. Ce qu'on veut, c'est simuler le temps d'une exécution de la tâche.

Pour cela, on va, dans une boucle, vérifier la différence entre le temps d'exécution actuel de la tâche (avec le champ *exectime* de `RT_TASK_INFO` mis à jour) et le temps d'exécution de la tâche initial (qu'on a appelé *begin*). Dès que cette différence atteint la durée voulue (durée d'exécution de la tâche donnée en paramètre), on peut sortir de la boucle et de la fonction. De cette façon, on a réalisé une attente active pendant la durée recherchée.

1.4 Question 4

On se sert de la fonction `time_since_start` dans la fonction `rt_task` pour avoir un point de repère temporel :

```
rt_printf("doing %s      time : %d\n",rt_task_name(), time_since_start
());
```

Voici le résultat de l'exécution du programme :

```
root@devkit8600-xenomai:~# ./pathfinder
started task ORDO_BUS, period 125ms, duration 25ms, use resource 0
doing ORDO_BUS    time : 125
doing ORDO_BUS ok  time : 150
doing ORDO_BUS    time : 250
doing ORDO_BUS ok  time : 275
doing ORDO_BUS    time : 375
doing ORDO_BUS ok  time : 400
doing ORDO_BUS    time : 500
doing ORDO_BUS ok  time : 525
doing ORDO_BUS    time : 625
doing ORDO_BUS ok  time : 650
doing ORDO_BUS    time : 750
doing ORDO_BUS ok  time : 775
doing ORDO_BUS    time : 875
doing ORDO_BUS ok  time : 900
doing ORDO_BUS    time : 1000
doing ORDO_BUS ok  time : 1025
doing ORDO_BUS    time : 1125
doing ORDO_BUS ok  time : 1150
doing ORDO_BUS    time : 1250
doing ORDO_BUS ok  time : 1275
```

FIGURE 4.1 – Résultat avec ORDO_BUS

Le timing est bon : 25ms d'exécution et 125ms de période.

1.5 Question 5

Pour une bonne coordination des tâches, le sémaphore doit être utilisé comme suit :

- Faire un `sem_p` (-1) dans `acquire_resource` et un `sem_v` (+1) dans `release_resource` sur le sémaphore
- Initialisation du sémaphore à 0 (bloque tout)
- Initialisation de toutes les tâches (et donc toutes les tâches sont bloquées)
- Faire un `sem_v` (+1) sur le sémaphore juste après ces initialisations (ce qui débloquent la tâche la plus prioritaire)

De cette façon, les tâches s'enchaîneront de la bonne manière selon leurs priorités.


```

doing METEO      time : 5226
doing ORDO_BUS   time : 5250
doing ORDO_BUS ok time : 5275
doing RADIO      time : 5275
doing RADIO ok   time : 5300
doing CAMERA     time : 5300
doing CAMERA ok  time : 5325
doing METEO ok   time : 5341
doing DISTRIB_DONNEES time : 5341
doing DISTRIB_DONNEES ok time : 5366
doing PILOTAGE   time : 5366
doing ORDO_BUS   time : 5375
doing ORDO_BUS ok time : 5400
doing PILOTAGE ok time : 5400
doing DISTRIB_DONNEES time : 5400
doing DISTRIB_DONNEES ok time : 5425
doing ORDO_BUS   time : 5500
doing ORDO_BUS ok time : 5525

```

FIGURE 4.2 – Enchaînement avec meilleur cas pour METEO (40ms)

```

doing METEO      time : 5226
doing ORDO_BUS   time : 5250
doing ORDO_BUS ok time : 5275
doing RADIO      time : 5275
doing RADIO ok   time : 5300
doing CAMERA     time : 5300
doing CAMERA ok  time : 5325
doing METEO ok   time : 5361
doing DISTRIB_DONNEES time : 5361
doing ORDO_BUS   time : 5375
doing ORDO_BUS ok time : 5400
doing DISTRIB_DONNEES ok time : 5411
doing PILOTAGE   time : 5411
doing PILOTAGE ok time : 5436
doing DISTRIB_DONNEES time : 5436
doing DISTRIB_DONNEES ok time : 5461
doing ORDO_BUS   time : 5500
doing ORDO_BUS ok time : 5525

```

FIGURE 4.3 – Enchaînement avec pire cas pour METEO (60ms)

On se rend compte que, lors du meilleur cas pour la tâche METEO (soit 40ms), ORDO_BUS n'a pas encore terminé sa période et n'est donc pas prêt pour exécution avant le début de la tâche PILOTAGE ou la fin de DISTRIB_DONNEES. Dans le pire cas pour la tâche METEO (soit 60ms), ORDO_BUS a atteint sa période et est donc exécuté avant la tâche PILOTAGE et s'intercale entre le début et le fin de l'exécution de DISTRIB_DONNEES.

1.6 Question 6

Messages de sortie du terminal entre l'allumage de la cible et le prompt de login

```

1 CCCCCCCC
  U-Boot SPL 2011.09-svn (May 22 2012 - 11:19:00)
3 Texas Instruments Revision detection unimplemented
  Booting from NAND...
5
7 U-Boot 2011.09-svn (May 22 2012 - 11:19:00)
9 I2C:   ready
  DRAM:  512 MiB
11 WARNING: Caches not enabled
  Did not find a recognized configuration , assuming General purpose EVM
    in Profile 0 with Daughter board
13 NAND:  HW ECC Hamming Code selected
    512 MiB
15 MMC:   OMAP SD/MMC: 0
  Net:    cpsw
17 Hit any key to stop autoboot:  3 \0x08\0x08\0x08 2 \0x08\0x08\0x08 1
    \0x08\0x08\0x08 0
  Card did not respond to voltage select!
19 Booting from network ...
  miiphy read id fail
21 link up on port 0, speed 100, full duplex
  BOOTP broadcast 1
23 DHCP client bound to address 192.168.1.6
  Using cpsw device
25 TFTP from server 192.168.1.1; our IP address is 192.168.1.6
  Filename 'uImage'.
27 Load address: 0x82000000
  Loading: *\0x08##
    #####
29 \0x09 ##
    #####
  \0x09 ##
    #####
31 \0x09 ##
    #####

```

```

\0x09 ##
#####
33 \0x09 ##
#####
\0x09 ##
#####
35 \0x09 ##
#####
\0x09 ##
#####
37 \0x09 #####
done
39 Bytes transferred = 3215152 (310f30 hex)
## Booting kernel from Legacy Image at 82000000 ...
41 Image Name: Linux-3.1.0
Image Type: ARM Linux Kernel Image (uncompressed)
43 Data Size: 3215088 Bytes = 3.1 MiB
Load Address: 80008000
45 Entry Point: 80008000
Verifying Checksum ... OK
47 Loading Kernel Image ... OK
OK
49
Starting kernel ...
51
Uncompressing Linux... done, booting the kernel.
53 Linux version 3.1.0 (mi11@mi11-VirtualBox) (gcc version 4.9.1 (GCC) )
#1 Mon Apr 10 18:15:11 CEST 2017
CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c53c7d
55 CPU: VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine: am335xevm
57 Memory policy: ECC disabled, Data cache writeback
AM335X ES1.0 (neon )
59 Built 1 zonelists in Zone order, mobility grouping on. Total pages:
130048
Kernel command line: console=ttyO0,115200n8 ip=dhcp noinitrd root=/
dev/nfs rw rootwait
61 PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
63 Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 512MB = 512MB total
65 Memory: 512996k/512996k available, 11292k reserved, 0K highmem
Virtual kernel memory layout:
67 vector : 0xffff0000 - 0xffff1000 ( 4 kB)
fixmap : 0xffff0000 - 0xffffe000 ( 896 kB)
69 DMA : 0xffa00000 - 0xffe00000 ( 4 MB)
vmalloc : 0xe0800000 - 0xf8000000 ( 376 MB)
71 lowmem : 0xc0000000 - 0xe0000000 ( 512 MB)
modules : 0xbf000000 - 0xc0000000 ( 16 MB)

```

```

73      .text : 0xc0008000 - 0xc05c6000    (5880 kB)
74      .init : 0xc05c6000 - 0xc05ff000    ( 228 kB)
75      .data : 0xc0600000 - 0xc065e618    ( 378 kB)
76      .bss : 0xc065e63c - 0xc0699694    ( 237 kB)
77 NR_IRQS:396
IRQ: Found an INTC at 0xfa200000 (revision 5.0) with 128 interrupts
79 Total of 128 interrupts on 1 active controller
OMAP clockevent source: GPTIMER1 at 25000000 Hz
81 OMAP clocksource: GPTIMER2 at 25000000 Hz
sched_clock: 32 bits at 25MHz, resolution 40ns, wraps every 171798ms
83 Console: colour dummy device 80x30
Calibrating delay loop... 718.02 BogoMIPS (lpj=3590144)
85 pid_max: default: 32768 minimum: 301
Security Framework initialized
87 Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
89 devtmpfs: initialized
print_constraints: dummy:
91 NET: Registered protocol family 16
GPMC revision 6.0
93 OMAP GPIO hardware version 0.1
omap_l3_smx omap_l3_smx.0: couldn't find resource
95 omap_mux_init: Add partition: #1: core, flags: 0
omap_i2c.1: alias fck already exists
97 The board is general purpose EVM in profile 0
omap_hsmmc.0: alias fck already exists
99 omap_hsmmc.2: alias fck already exists
Configure Bluetooth Enable pin...
101 error setting wl12xx data
omap2_mcspi.1: alias fck already exists
103 omap2_mcspi.2: alias fck already exists
bio: create slab <bio-0> at 0
105 SCSI subsystem initialized
usbcore: registered new interface driver usbfs
107 usbcore: registered new interface driver hub
usbcore: registered new device driver usb
109 registerd cppi-dma Intr @ IRQ 17
Cppi41 Init Done Qmgr-base(e083a000) dma-base(e0838000)
111 Cppi41 Init Done
omap_i2c omap_i2c.1: bus 1 rev4.0 at 100 kHz
113 Advanced Linux Sound Architecture Driver Version 1.0.24.
Bluetooth: Core ver 2.16
115 NET: Registered protocol family 31
Bluetooth: HCI device and connection manager initialized
117 Bluetooth: HCI socket layer initialized
Bluetooth: L2CAP socket layer initialized
119 Bluetooth: SCO socket layer initialized
Switching to clocksource gp timer
121 Switched to NOHz mode on CPU #0

```

```

musb-hdrc: version 6.0, ?dma?, otg (peripheral+host)
123 musb-hdrc musb-hdrc.0: dma type: dma-cppi41
musb-hdrc musb-hdrc.0: USB OTG mode controller at e080a000 using DMA,
    IRQ 18
125 musb-hdrc musb-hdrc.1: dma type: dma-cppi41
musb-hdrc musb-hdrc.1: USB OTG mode controller at e080c800 using DMA,
    IRQ 19
127 NET: Registered protocol family 2
IP route cache hash table entries: 4096 (order: 2, 16384 bytes)
129 TCP established hash table entries: 16384 (order: 5, 131072 bytes)
TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
131 TCP: Hash tables configured (established 16384 bind 16384)
TCP reno registered
133 UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
135 NET: Registered protocol family 1
RPC: Registered named UNIX socket transport module.
137 RPC: Registered udp transport module.
RPC: Registered tcp transport module.
139 RPC: Registered tcp NFSv4.1 backchannel transport module.
NetWinder Floating Point Emulator V0.97 (double precision)
141 VFS: Disk quotas dquot_6.5.2
Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
143 JFFS2 version 2.2. (NAND) (SUMMARY) \0xc2\0xa9 2001-2006 Red Hat,
    Inc.
msgmni has been set to 1001
145 io scheduler noop registered
io scheduler deadline registered
147 io scheduler cfq registered (default)
Could not set LED4 to fully on
149 da8xx_lcd dc da8xx_lcd.0: GLCD: Found AT043TN24 panel
Console: switching to colour frame buffer device 60x34
151 Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
omap_uart.0: ttyO0 at MMIO 0x44e09000 (irq = 72) is a OMAP UART0
153 console [ttyO0] enabled
omap_uart.1: ttyO1 at MMIO 0x48022000 (irq = 73) is a OMAP UART1
155 omap_uart.2: ttyO2 at MMIO 0x48024000 (irq = 74) is a OMAP UART2
omap_uart.3: ttyO3 at MMIO 0x481a6000 (irq = 44) is a OMAP UART3
157 omap_uart.4: ttyO4 at MMIO 0x481a8000 (irq = 45) is a OMAP UART4
omap_uart.5: ttyO5 at MMIO 0x481aa000 (irq = 46) is a OMAP UART5
159 brd: module loaded
loop: module loaded
161 i2c-core: driver [tsl2550] using legacy suspend method
i2c-core: driver [tsl2550] using legacy resume method
163 mtdoops: mtd device (mtddev=name/number) must be supplied
omap2-nand driver initializing
165 ONFI flash detected
ONFI param page 0 valid

```

```

167 NAND device: Manufacturer ID: 0xad, Chip ID: 0xdc (Hynix H27U4G8F2DTR
    -BC)
    Creating 8 MID partitions on "omap2-nand.0":
169 0x000000000000-0x000000020000 : "SPL"
    0x000000020000-0x000000040000 : "SPL.backup1"
171 0x000000040000-0x000000060000 : "SPL.backup2"
    0x000000060000-0x000000080000 : "SPL.backup3"
173 0x000000080000-0x0000000260000 : "U-Boot"
    0x0000000260000-0x0000000280000 : "U-Boot Env"
175 0x0000000280000-0x0000000780000 : "Kernel"
    0x0000000780000-0x00000020000000 : "File System"
177 OneNAND driver initializing
    davinci_mdio davinci_mdio.0: davinci mdio revision 1.6
179 davinci_mdio davinci_mdio.0: detected phy mask ffffffff
    davinci_mdio.0: probed
181 davinci_mdio davinci_mdio.0: phy[4]: device 0:04, driver unknown
    CAN device driver interface
183 CAN bus driver for Bosch D_CAN controller 1.0
    d_can d_can: d_can device registered (irq=55, irq_obj=56)
185 usbcore: registered new interface driver cdc_ether
    usbcore: registered new interface driver cdc_subset
187 Initializing USB Mass Storage driver...
    usbcore: registered new interface driver usb-storage
189 USB Mass Storage support registered.
    gadget: using random self ethernet address
191 gadget: using random host ethernet address
    usb0: MAC 06:50:55:8c:0c:93
193 usb0: HOST MAC 62:9a:93:a2:1e:53
    gadget: Ethernet Gadget, version: Memorial Day 2008
195 gadget: g_ether ready
    musb-hdrc musb-hdrc.0: MUSB HDRC host driver
197 musb-hdrc musb-hdrc.0: new USB bus registered, assigned bus number 1
    usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
199 usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
    usb usb1: Product: MUSB HDRC host driver
201 usb usb1: Manufacturer: Linux 3.1.0 musb-hcd
    usb usb1: SerialNumber: musb-hdrc.0
203 hub 1-0:1.0: USB hub found
    hub 1-0:1.0: 1 port detected
205 musb-hdrc musb-hdrc.1: MUSB HDRC host driver
    musb-hdrc musb-hdrc.1: new USB bus registered, assigned bus number 2
207 usb usb2: New USB device found, idVendor=1d6b, idProduct=0002
    usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
209 usb usb2: Product: MUSB HDRC host driver
    usb usb2: Manufacturer: Linux 3.1.0 musb-hcd
211 usb usb2: SerialNumber: musb-hdrc.1
    hub 2-0:1.0: USB hub found
213 hub 2-0:1.0: 1 port detected
    mousedev: PS/2 mouse device common for all mice

```

```

215 input: ti-tsc-adcc as /devices/platform/tsc/input/input0
omap_rtc omap_rtc: rtc core: registered omap_rtc as rtc0
217 i2c /dev entries driver
Linux video capture interface: v2.00
219 usbcore: registered new interface driver uvcvideo
USB Video Class driver (1.1.1)
221 OMAP Watchdog Timer Rev 0x01: initial timeout 60 sec
Bluetooth: HCI UART driver ver 2.2
223 Bluetooth: HCI H4 protocol initialized
Bluetooth: HCI BCSP protocol initialized
225 Bluetooth: HCILL protocol initialized
Bluetooth: HCIATH3K protocol initialized
227 cpuidle: using governor ladder
cpuidle: using governor menu
229 usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
231 usbcore: registered new interface driver snd-usb-audio
_regulator_get: 1-000a supply VDDA not found, using dummy regulator
233 _regulator_get: 1-000a supply VDDIO not found, using dummy regulator
_regulator_get: 1-000a supply VDDD not found, using dummy regulator
235 sgtl5000 1-000a: sgtl5000 revision 17
print_constraints: 1-000a: 850 <=> 1600 mV at 1200 mV normal
237 _regulator_get: 1-000a supply VDDA not found, using dummy regulator
_regulator_get: 1-000a supply VDDIO not found, using dummy regulator
239 sgtl5000 1-000a: Using internal LDO instead of VDDD
mmc1: card claims to support voltages below the defined range. These
will be ignored.
241 sgtl5000 1-000a: Failed to add route HPLOUT->Headphone Jack
sgtl5000 1-000a: dapm: unknown pin MONO_LOUT
243 sgtl5000 1-000a: dapm: unknown pin HPLCOM
sgtl5000 1-000a: dapm: unknown pin HPRCOM
245 asoc: sgtl5000 <-> davinci-mcasp.0 mapping ok
ALSA device list:
247 #0: AM335X EVM
oprofile: hardware counters not available
249oprofile: using timer interrupt.
nf_connttrack version 0.5.0 (8015 buckets, 32060 max)
251 ip_tables: (C) 2000-2006 Netfilter Core Team
TCP cubic registered
253 NET: Registered protocol family 17
can: controller area network core (rev 20090105 abi 8)
255 NET: Registered protocol family 29
can: raw protocol (rev 20090105)
257 can: broadcast manager protocol (rev 20090105 t)
Bluetooth: RFCOMM TTY layer initialized
259 Bluetooth: RFCOMM socket layer initialized
Bluetooth: RFCOMM ver 1.11
261 Bluetooth: BNEP (Ethernet Emulation) ver 1.3
Bluetooth: BNEP filters: protocol multicast

```



```

263 Bluetooth: HIDP (Human Interface Emulation) ver 1.2
    Registering the dns_resolver key type
265 VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev
    3
    ThumbEE CPU extension supported.
267 _regulator_get: mpu.0 supply mpu not found, using dummy regulator
omap2_set_init_voltage: Fail set voltage-dpll_mpu_ck(f=720000000 v
    =1260000)on vddmpu
269 omap2_set_init_voltage: unable to set vdd_mpu
    Detected MACID=0:18:30:fe:7b:f6
271 input: gpio-keys as /devices/platform/gpio-keys/input/input1
omap_rtc omap_rtc: setting system clock to 2000-01-01 00:00:00 UTC
    (946684800)
273 mmc1: queuing unknown CIS tuple 0x91 (3 bytes)

275 CPSW phy found : id is : 0x4dd072
    PHY 0:01 not found
277 mmc1: new SDIO card at address 0001
    PHY: 0:04 - Link is Up - 100/Full
279 Sending DHCP requests ., OK
    IP-Config: Got DHCP answer from 192.168.1.1, my address is
    192.168.1.6
281 IP-Config: Complete:
    device=eth0, addr=192.168.1.6, mask=255.255.255.0, gw
    =255.255.255.255,
283     host=192.168.1.6, domain=, nis-domain=(none),
    bootserver=192.168.1.1, rootserver=192.168.1.1, rootpath=/
    tftpboot/rootfs
285 VFS: Mounted root (nfs filesystem) on device 0:15.
    devtmpfs: mounted
287 Freeing init memory: 228K

289 INIT: version 2.88 booting

291 Starting udev
    udevd[718]: starting version 182
293 bootlogd: cannot allocate pseudo tty: No such file or directory
    Populating dev cache
295 Fri Apr 14 13:12:58 UTC 2017

297 INIT: Entering runlevel: 5

299 Configuring network interfaces... ifup skipped for nfsroot interface
    eth0
    run-parts: /etc/network/if-pre-up.d/nfsroot exited with code 1
301 Starting system message bus: dbus.
    Starting Dropbear SSH server: Generating key, this may take a while
    ...
303 Public key portion is:

```

```

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDHQW67zEKPRleKx6VZxLER0R/2HThSN
/
SWD3fMk23eXy9j3PudyVMJfjGBF258qnZNicoMsK0Mx5JrpV124XKCzvKTAYsMjLc67WdqcX73zSzt1Cpl
+Dq16Nld2ZuhfGDidLPvrSqOfRaUYRH6048XV/
E7penoQ8oP2tJV0kiGnXRQMoqSyLyZFWW/0xNzatB/Wr6o+
I7Iboc2KWDyOu8caJxP4fxrV/4zEjyTvSQCBCzwuv2RSFPJV7lleMD2XkKN+
QOGCgG1Eq8TgrNDU0Jps+RelENrtkj+lgVJF2odabmPMtmsB+8
lhtgdgk8wth0dbjQzedRFt root@devkit8600
305 Fingerprint: md5 a7:0c:a2:b1:07:9a:77:98:01:7e:31:13:10:02:42:2c
dropbear.
307 Starting rpcbind daemon...rpcbind: cannot create socket for udp6

309 rpcbind: cannot create socket for tcp6

311 done.
Starting syslogd/klogd: done
313 * Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
...done.
315 Starting Telephony daemon
Starting Linux NFC daemon
317 /etc/rc5.d/S64neard: line 26: /usr/lib/neard/nfc/neard: No such file
or directory

319 Poky (Yocto Project Reference Distro) 1.7.3 devkit8600 /dev/ttyO0
321
323 devkit8600 login: root
325 root@devkit8600:~#

```

Listing A.1 – Messages de sortie du terminal entre l’allumage de la cible et le prompt de login